UNIVERSITAT JAUME I

Máster Universitario en Diseño y Fabricación

SISTEMA DE ACOPLAMIENTO Y ACCIONAMIENTO DE MANOS ARTIFICIALES PARA UN ROBOT MANIPULADOR

TRABAJO FIN DE MÁSTER

AUTOR
Adrián Martínez Márquez
TUTORA
Marta Covadonga Mora Aguilar

Castellón, diciembre 2019

ÍNDICE GENERAL

1. MEMORIA

ÍNDICE

1 INTRODUCCIÓN

Existen multitud de dificultades técnicas en el desarrollo de manos protésicas tanto en el diseño como en la adquisición de datos mediante sensores e incluso en el control. Todo esto conlleva un elevado coste final del producto. Además, no existe una metodología clara para evaluar la capacidad de manipulación de una mano artificial específica.

Por este motivo, el grupo de Biomecánica y Ergonomía de la Universitat Jaume I ha estado trabajando en esta linea de investigación en el marco de dos proyectos nacionales denominados DEVALHAND [1] y BENCH-HAND [2], los cuales tienen dos objetivos, el primero consiste en desarrollar metodologías que permitan evaluar y optimizar las capacidades de los distintos diseños de manos protésicas. Y segundo, partiendo de las metodologías halladas, desarrollar nuevos diseños de bajo coste.

Para lograr estos objetivos en los proyectos, se parte de la hipótesis de que el uso intensivo de herramientas de simulación previo al desarrollo del prototipo físico conseguirá mejorar las etapas de diseño, evaluación y control. Esto junto con la aplicación de técnicas de prototipado rápido permitirá la reducción de los costes.

1.1 Justificación

El trabajo se enmarca en la línea de investigación en biomecánica de la mano que desarrolla el grupo de Biomecánica y Ergonomía de la Universitat Jaume I.

Como resultado del proyecto DEVALHAND se han obtenido dos manos protésicas llamadas *IMMA Hand* [3] y *BruJa Hand* [4]. Actualmente se dispone de varios dispositivos de medida para estudiar el desempeño de estas manos. No obstante, se carece de una forma de analizar el comportamiento contemplando también el movimiento del brazo. Por este motivo desde grupo de Biomecánica y Ergonomía de la Universitat Jaume I se decide adquirir el brazo robótico educacional *Dobot Magician* [5]. Acoplando las manos existentes a este robot se podrá simular también posiciones y movimientos de un brazo a la vez que se actúan las manos.

1.2 Objetivos

Los objetivos principales del presente trabajo son:

- Montaje y puesta en marcha del robot manipulador *Dobot Magician.*
- Diseñar un sistema de acoplamiento de manos artificiales para dicho robot.
- Realizar el control de posición de la mano tanto de forma real como simulada.

1.3 Alcance

El alcance de este trabajo incluye el diseño, fabricación y montaje de todas las piezas necesarias para poder acoplar manos protésicas al brazo robot *Dobot Magician.* También incluye la programación del control del brazo para poder situarlo en distintas posiciones con la intención de realizar futuras pruebas o ensayos con las manos acopladas.

Las trayectorias y movimientos obtenidos tras aplicar el control deberán poder ser simulados con antelación de forma virtual para poder verificar que son correctos antes de que el robot los ejecute.

El control del movimiento de los dedos de las manos para realizar los agarres está fuera del alcance de este trabajo.

2 ANTECEDENTES

2.1 Análisis del producto

Para el trabajo se utilizará el brazo robótico educacional *Dobot Magician*. Por tanto, es necesario conocer las dimensiones de este y el rango de movimiento del que dispone para evitar posibles interferencias o colisiones durante su movimiento [6]. También se analizan las posibles formas de control que ofrece el fabricante.

2.1.1 Características

El robot consta de 4 eslabones además de la base. Para referirse a estos, se han denominado, por analogía con el cuerpo humano: hombro, brazo, antebrazo y muñeca.

La denominación de cada eslabón puede verse en la Figura 1, junto con las dimensiones generales.

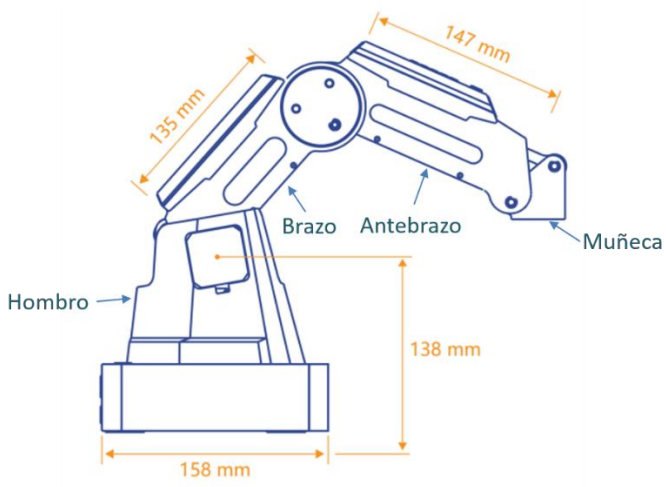


Figura 1. Denominación de los eslabones y dimensiones generales de Dobot Magician

En cuanto al movimiento del brazo, este consta de 4 grados de libertad correspondientes cada una de las articulaciones (J1, J2, J3 y J4).

El fabricante define el ángulo girado de cada articulación según se muestra en la Figura 2. El ángulo girado por la articulación J3 no se define con respecto al anterior eslabón como es habitual en el control de robots, sino que se mide con respecto a la horizontal.

El eje de rotación J4 depende del módulo instalado en el robot, por lo tanto, se considera que la estructura del brazo solo posee 3 grados de libertad. Por ello, no se dispone de grados de libertad en la zona correspondiente a la muñeca. No obstante, debido al mecanismo interno del brazo, el acoplador existente en la zona de la muñeca gira respecto a un eje paralelo al eje J3, de modo que mantiene su orientación respecto del plano horizontal.

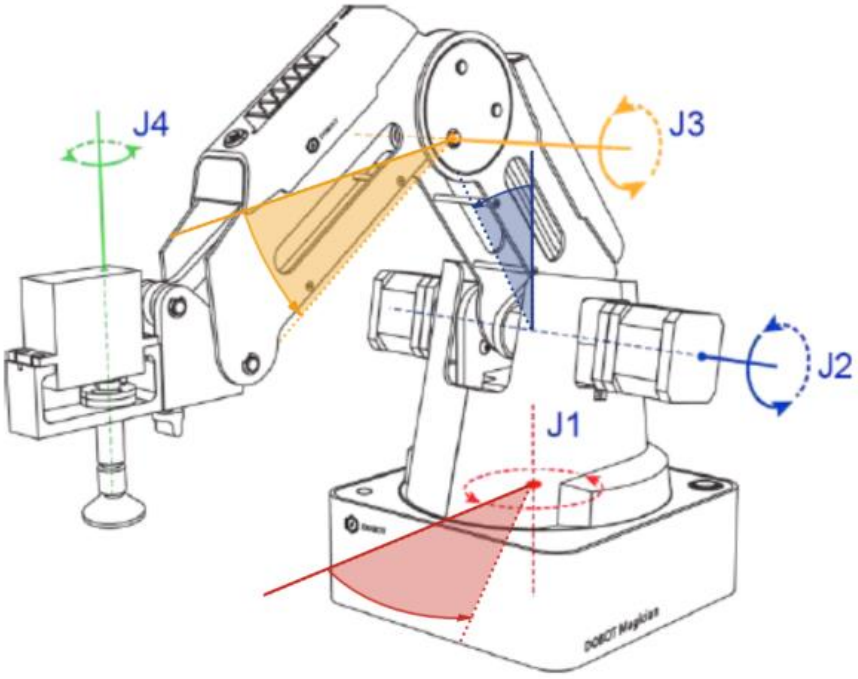


Figura 2. Ejes de rotación de Dobot Magician

Este mismo acoplador utilizado para fijar los distintos módulos del robot se utilizará para sujetar las manos, de modo que no se requiera ninguna modificación de las piezas del robot. Las dimensiones del acoplador pueden verse en la Figura 3.

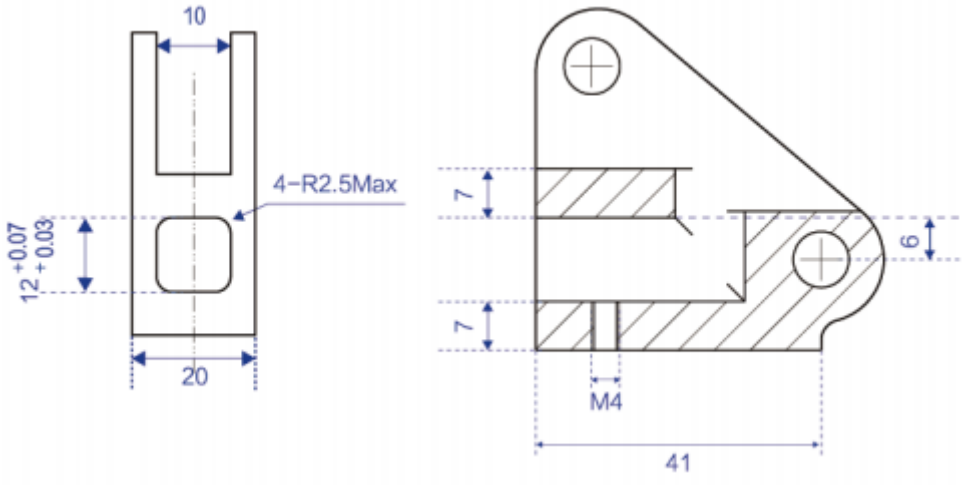


Figura 3. Dimensiones acoplador de Dobot Magician

El acoplamiento de los módulos se realiza simplemente a través de un saliente que se introduce en el acoplador. Posteriormente, se aprisiona mediante un tornillo de cabeza en mariposa como se muestra en la Figura 4.

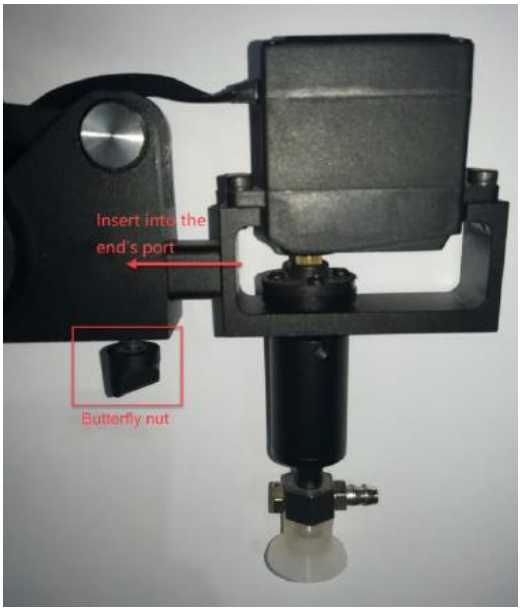


Figura 4. Acoplamiento del módulo de ventosa Dobot Magician

Además de conocer la geometría y movimiento del brazo también es necesario conocer características como la carga máxima, la velocidad máxima o el rango de movimiento de cada eje. Un resumen de todas estas características puede verse en la Tabla 2.1 [8].

Tabla 2.1. Características de Dobot Magician

Carga máxima	500g	
Alcance máximo	320 mm	
Rango de movimiento	Base	-90° a 90°
	Brazo	0° a 85°
	Antebrazo	-10° a 90°
	Rotación del efector final	-135° a 135°
Máxima velocidad (Carga: 250 g)	Base, brazo y antebrazo	320°/s
	Servo	480°/s
Precisión de posicionamiento	0,2 mm	

2.1.2 Control

En primer lugar, el movimiento del robot puede definirse en un sistema de coordenadas cartesiano o en un sistema de coordenadas por articulaciones. En el primero se define la posición del actuador y en el segundo se especifica el ángulo de cada articulación (ver Figura 5).

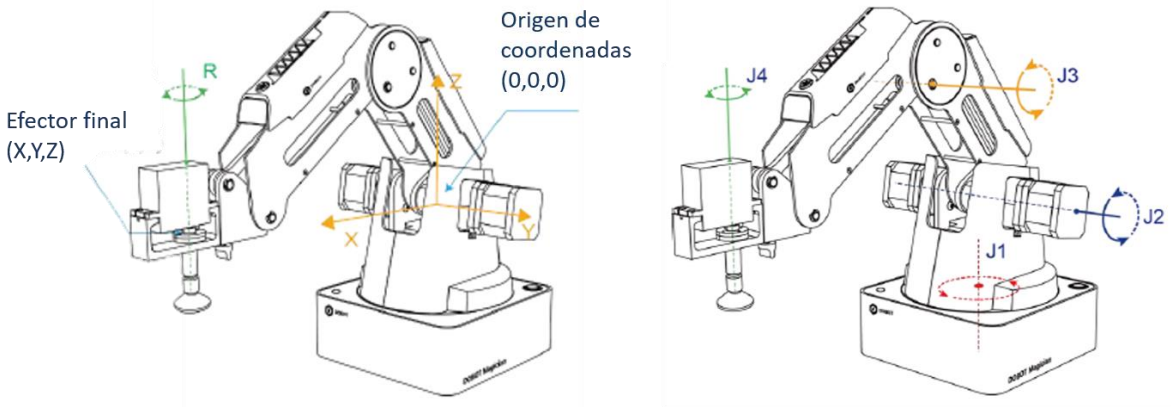


Figura 5. Sistema de coordenadas cartesiano (izquierda) y sistema de coordenadas por articulaciones (derecha)

El control del robot se realiza mediante el envío de comandos desde otro dispositivo (PC, Arduino, etc.). Mediante estos comandos se especifica el modo de desplazamiento y los parámetros necesarios para el movimiento del robot en dicho modo. El controlador del robot procesa esta información y acciona los actuadores para realizar el movimiento indicado (Figura 6).

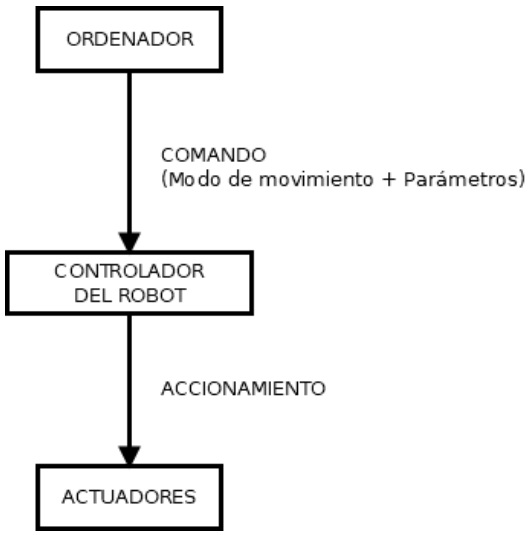


Figura 6. Esquema de control del robot

El robot presenta varios modos de control del movimiento. Estos modos se pueden configurar desde la aplicación proporcionada por el fabricante o bien mediante el uso de funciones de una librería también proporcionada por el fabricante.

El fabricante denomina estos modos de control: PTP, ARC, JOG y CP.

- PTP

El modo PTP consiste en el movimiento del brazo punto a punto. A su vez este modo tiene varias formas de operar denominadas MOVJ, MOVL y JUMP.

En el modo MOVJ el movimiento desde el punto inicial al final se realiza sin tener en cuenta ninguna trayectoria específica del efector final como se muestra en la Figura 7.

Al utilizar el modo MOVL el movimiento desde el punto inicial hasta el final se realiza de forma rectilínea como se muestra en la Figura 7.

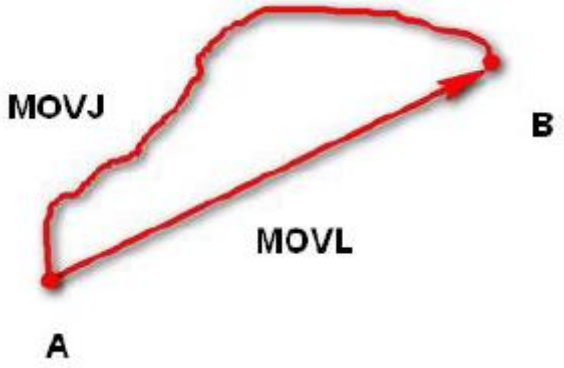


Figura 7. Modos MOVJ y MOVL

En el modo JUMP el actuador del brazo primero sube una distancia vertical determinada, posteriormente se desplaza horizontalmente hasta situarse encima del punto final y finalmente baja hasta dicho punto. (ver Figura 8)

Figura 8. Modo JUMP

Además, cada uno de estos modos de movimiento PTP se puede configurar de varias formas.

- JUMP_XYZ: Modo JUMP. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas cartesiano.
- MOVJ_XYZ: Modo MOVJ. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas cartesiano.

- MOVL_XYZ: Modo MOVL. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas cartesiano.
- JUMP_ANGLE: Modo JUMP. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas articular.
- MOVJ_ANGLE: Modo MOVJ. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas articular.
- MOVL_ANGLE: Modo MOVL. Las coordenadas (x,y,z,r) especificadas se interpretan como el punto de destino en el sistema de coordenadas articular.
- MOVJ_INC: Modo MOVJ. Las coordenadas (x,y,z,r) especificadas se interpretan como el incremento de ángulo en el sistema de coordenadas articular.
- MOVL_INC: Modo MOVL. Las coordenadas (x,y,z,r) especificadas se interpretan como el incremento de ángulo en el sistema de coordenadas articular.
- MOVJ_XYZ_INC: Modo MOVJ. Las coordenadas (x,y,z,r) especificadas se interpretan como el incremento de posición en el sistema de coordenadas cartesiano.
- JUMP_MOVL_XYZ: Modo JUMP. Las coordenadas (x,y,z,r) especificadas se interpretan como el incremento de posición en el sistema de coordenadas cartesiano.

- ARC

En este modo la trayectoria describe un arco definido por tres puntos según se muestra en la Figura 9.

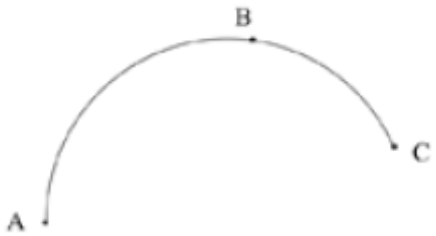


Figura 9. Modo ARC

- JOG

Utilizando este modo el robot se mueve por pasos de forma incremental ya sea en el sistema de coordenadas cartesiano o el sistema de coordenadas por articulaciones.

- CP (Trayectoria continua)

En este modo el controlador tiene en cuenta los siguientes comandos en la cola de comandos. De esta forma, los perfiles de aceleración y velocidad de la trayectoria seguida por el efector final tienen una mayor continuidad y menos saltos bruscos.

Este modo únicamente permite definir el punto de destino del efector final en el sistema de coordenadas cartesiano.

2.2 Análisis de mercado

2.2.1 Tipos de manos existentes

Para el diseño del acoplador se contemplan dos tipos de manos artificiales: las basadas en tendones con actuadores en el exterior de la mano y las basadas en mecanismos articulados o mixtos que contengan los actuadores en el interior.

Las manos que utilizan actuadores situados fuera de la misma se valen de hilos o cuerdas para transmitir el movimiento a los dedos. Esta situación plantea una serie de problemas a la hora de diseñar el acoplador. Primero, se necesita un soporte para alojar estos actuadores y segundo, la posición relativa entre los dedos y los actuadores influirá en la posición de los primeros por lo que esto se deberá tener en cuenta al diseñar el soporte y en el control de la mano si fuese necesario. Además, debe tenerse en cuenta el espacio necesario para el paso de los tendones y las guías que se puedan necesitar. Un ejemplo de este tipo de mano se puede ver en la Figura 10.

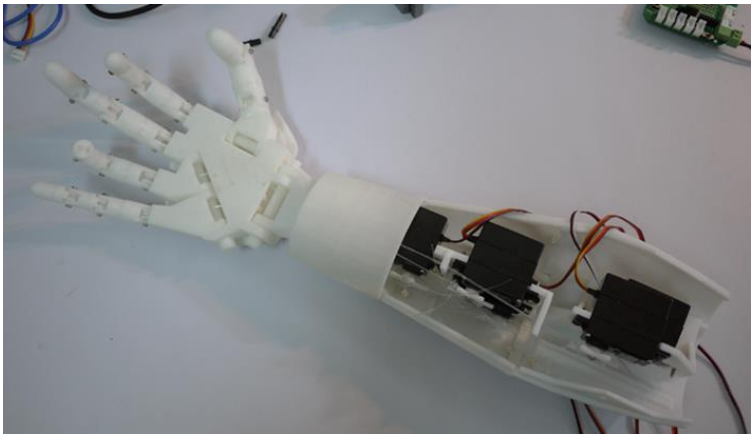


Figura 10. Mano robótica InMoov [7]

Por otro lado, en las manos que contienen los actuadores en el interior como se muestra en la Figura 11, se simplifica el diseño del acoplador, que no necesitará un soporte adicional para los actuadores.

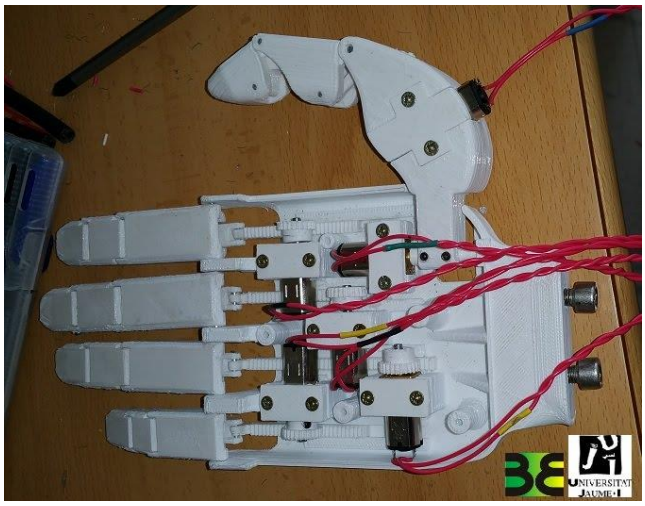


Figura 11. Mano basada en mecanismos articulados (BruJa Hand)

En la Figura 11 también puede observarse el sistema de sujeción existente en las manos desarrolladas por el grupo de Biomecánica y Ergonomía de la Universitat Jaume I. Este consiste en dos agujeros roscados en la parte de la muñeca que son utilizados para unir la mano a un soporte mediante tornillos.

Existen diferentes tipos de manos protésicas en cuanto al número de actuadores y grados de libertad. Se pueden agrupar de la siguiente forma:

- Manos con un solo actuador. Generalmente utilizan un sistema de tendones para ello (normalmente 5 tendones) y son actuados mediante el movimiento de muñeca de la persona que lo utiliza.
- Varios actuadores (En el exterior de la mano).
- Varios actuadores (En el interior de la mano).

En cuanto al peso total del conjunto de la mano y los actuadores, existen diseños que pueden alcanzar valores cercanos a los 400g.

2.2.2 Actuadores

Para el control de las manos, tanto para las basadas en tendones como para las basadas en mecanismos articulados, se suelen utilizar servomotores (Figura 12) o motores de corriente continua con reductor (Figura 13).

Figura 12. Servomotor [9]

Figura 13. Motor de corriente continua con reductor [10]

3 DISEÑO CONCEPTUAL

3.1 Especificaciones de diseño

3.1.1 Piezas

E1. Que el sistema sea válido para el accionamiento de manos basadas en tendones y mecanismos articulados. (Restricción)

E2. Que sirva para acoplar las manos protésicas existentes IMMA y BruJa. (Restricción)

E3. Estaría bien que permitiese el guiado del cableado de alimentación, accionamiento y posible sensorización de las manos. (Deseo)

E4. Que se pueda fabricar mediante impresión 3D. (Restricción)

E5. Que las piezas pesen lo menos posible y que su masa sea inferior a 100 g. (Optimizable)

E6. Que sea lo más barata posible. (Optimizable)

E7. Que las uniones soporten como mínimo 500 g (Esta especificación solo se aplica para la unión de la mano con el brazo). (Restricción)

E8. Que no interfiera con el movimiento del brazo robótico. (Restricción)

E9. Estaría bien que se pudiera montar y desmontar en menos de 15 minutos. (Deseo)

E10. Estaría bien que incluyese la rotación de la muñeca. (Solo se aplica a la unión de la mano con el brazo) (Deseo)

E11. Que puedan acoplarse la mayor cantidad de actuadores distintos posibles. (Sólo se aplica al soporte de los actuadores)

3.1.2 Control

C1. Que el control sea capaz de posicionar el efector final en un punto especificado. (Restricción)

C2. Que las trayectorias y posiciones del robot calculadas se puedan simular. (Restricción)

C3. Que permita conocer la posición del robot en cada instante. (Restricción)

C4. Que permita controlar la velocidad de movimiento en cada instante. (Restricción)

C5. Que se pueda implementar en diferentes plataformas o sistemas operativos. (Restricción)

C6. Que sea lo más barato posible. (Optimizable)

C7. Que sea lo más fácil de programar posible. (Optimizable)

3.2 Alternativas

3.2.1 Piezas

Se han realizado varios diseños de cada pieza a diseñar para elegir la mejor solución para cada una. Las piezas diseñadas son la unión del brazo con la mano, el soporte de los actuadores y el soporte de la electrónica.

- Unión del antebrazo con la mano

La primera propuesta se basa en la unión atornillando el tornillo con cabeza de mariposa que trae el brazo robot a una tuerca incrustada en la pieza. La función de la tuerca es aportar resistencia a la unión puesto que en principio el acoplador será de plástico. Esta propuesta puede verse en la Figura 14.

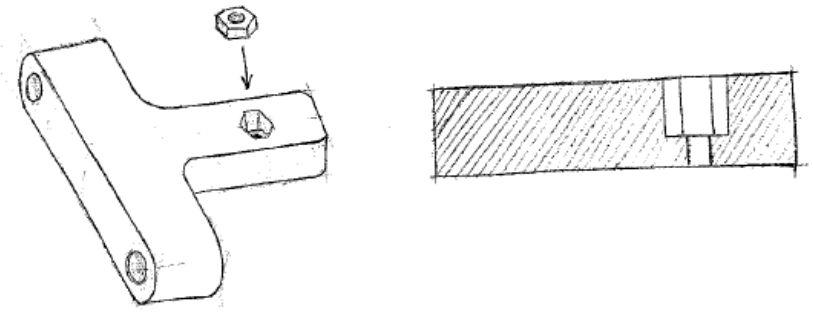


Figura 14. Propuesta 1 - Unión del antebrazo con la mano

Puesto que la unión con el antebrazo presenta complicaciones por la resistencia del plástico se ha ideado la propuesta 2, que sustituye esa parte del acoplador por un perfil metálico. A su vez este perfil iría unido a la parte de plástico mediante una abrazadera que lo sujeta por fricción como puede verse en la Figura 15.

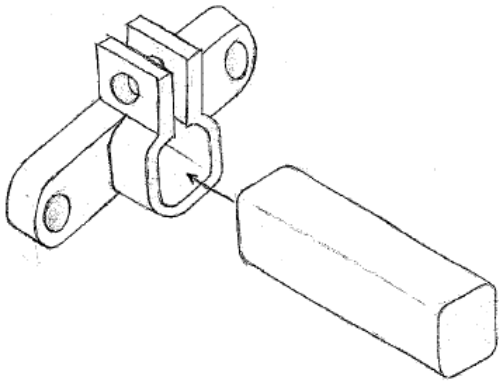


Figura 15. Propuesta 2 - Unión del antebrazo con la mano

También, se ha considerado el uso del acoplador que incluye el propio brazo robot, para lo cual sería necesario desmontar el servo que lleva incorporado. Esta pieza puede verse en la parte superior de la Figura 16. De este modo se soluciona el acople con el antebrazo. Para el acople de la

mano se propone la pieza que aparece en la parte inferior que irá unida mediante 4 tornillos a la otra.

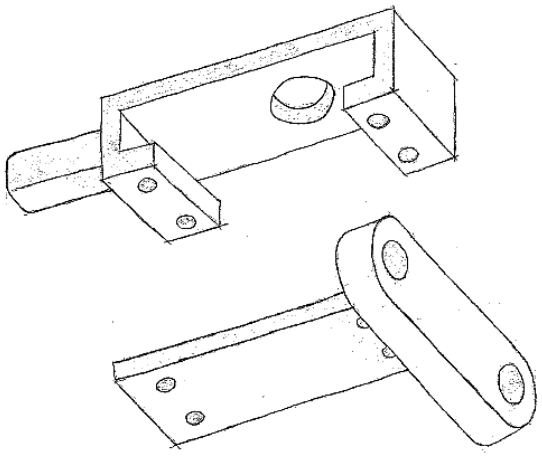


Figura 16. Propuesta 3 - Unión del antebrazo con la mano

Por último, se ha considerado la posibilidad de añadir un grado de libertad más, correspondiente al giro de la muñeca. Para ello, se hace uso del propio servomotor que incorpora el robot. Como ya se ha comentado anteriormente, dicho servomotor ya dispone de un acoplador para unirlo al brazo robot, no obstante, la orientación del eje de giro no es la adecuada y por tanto se necesita diseñar un soporte diferente. En la Figura 17 se muestra la cuarta propuesta que consiste en un soporte para el servo y un acoplador para sujetar la mano a dicho servo.

El acoplador de la mano tendría una zona con agujeros en la parte superior para colocar unos tubos que sirvan de guía en el caso de las manos con tendones. En caso de que se acople una mano sin tendones, se podrían utilizar estos agujeros para el guiado de los cables de alimentación.

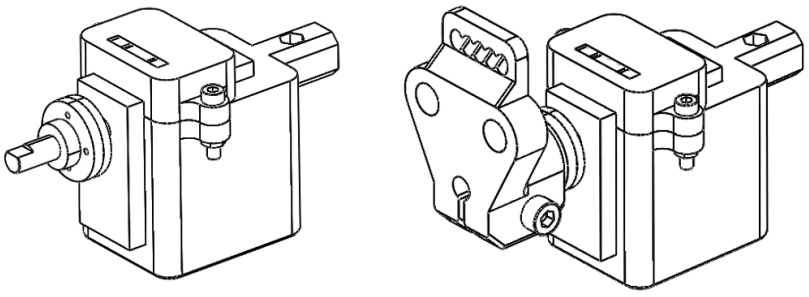


Figura 17. Propuesta 4 - Unión del antebrazo con la mano

- Soporte de los actuadores

Para el soporte de los actuadores se ha separado el diseño en dos partes, por un lado, se proponen varias alternativas para la sujeción de los actuadores y por otro se proponen varios sistemas de sujeción al antebrazo.

La primera propuesta para el soporte de los actuadores consiste en una base que será independiente del tipo o tamaño de actuador. Esta contará con agujeros para el ensamblaje de

otra pieza que será la que se adapte a los actuadores específicos. En la Figura 18 puede verse esta primera propuesta. En líneas discontinuas se representa donde se situaría el anclaje con el antebrazo, para el cual se proponen varias alternativas más adelante.

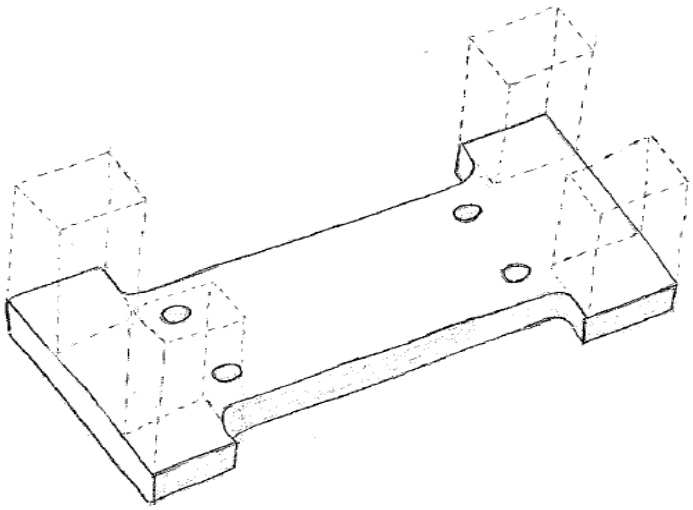


Figura 18. Propuesta 1 - Soporte de los actuadores

La segunda propuesta sería parecida a la primera pero esta vez incluyendo todo en una sola pieza de modo que se necesitaría un soporte para cada tipo de actuador que se quiera utilizar. Por otra parte, se tendría la ventaja de una pieza más sencilla y con un menor peso al no contar con tornillos para unir las dos partes. Un ejemplo de la segunda propuesta para el soporte de 5 servomotores puede verse en la Figura 19.

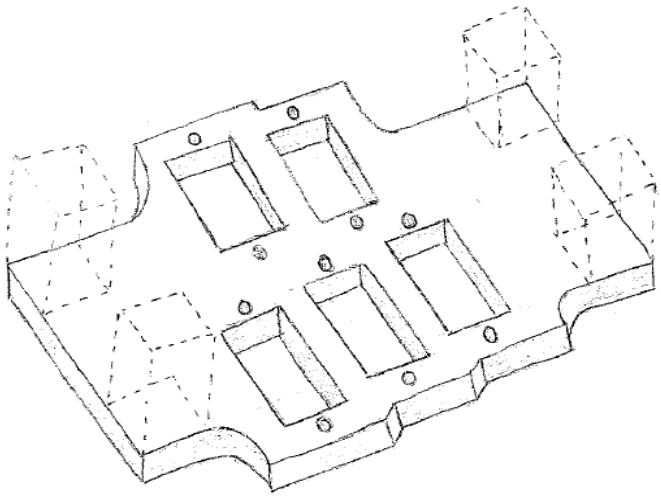


Figura 19. Propuesta 2 - Soporte de los actuadores

Una tercera propuesta consiste en utilizar ranuras en el soporte que permitan el anclaje de los actuadores. Estas ranuras aportarían una mayor flexibilidad. Esta alternativa puede verse en la Figura 20.

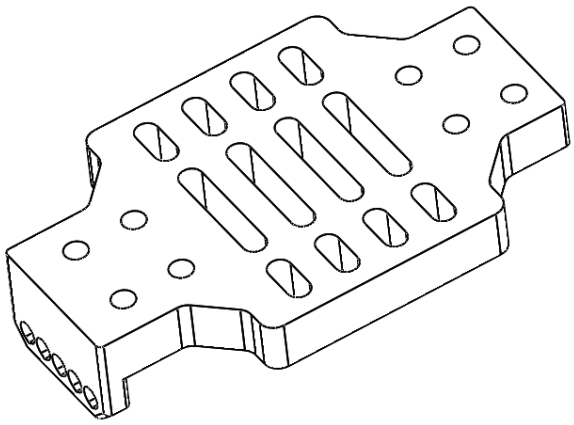


Figura 20. Propuesta 3 – Soporte de los actuadores

Para la unión del soporte con el antebrazo se proponen dos alternativas. La primera puede verse en la Figura 21 y consiste en un saliente con una ranura para encajar el perfil del antebrazo y un agujero por el que pasar una brida que servirá para asegurar la unión.

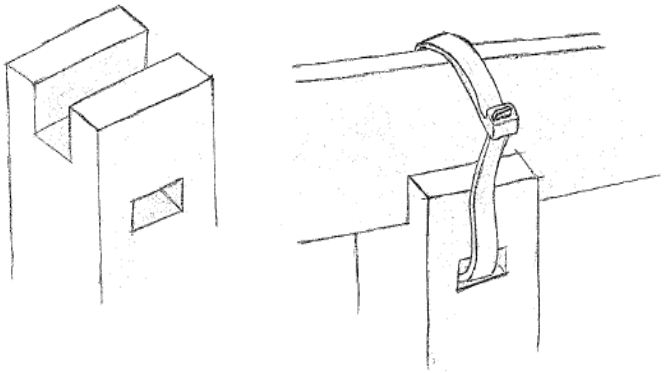


Figura 21. Propuesta 1 - Sistema de sujeción del soporte de los actuadores

Una segunda alternativa consiste en una abrazadera que rodee parte del perfil del antebrazo y se apriete mediante un tornillo tal y como se muestra en la Figura 22.

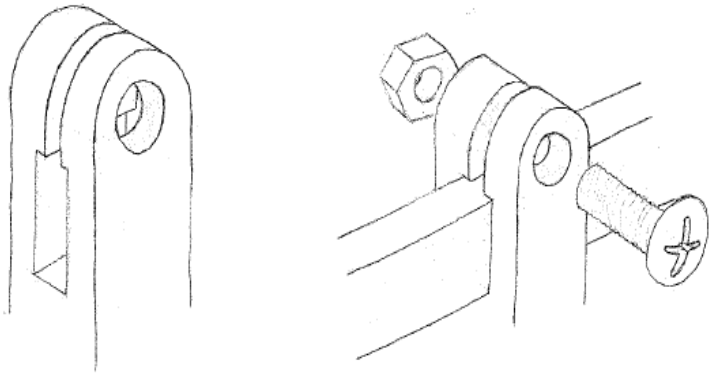


Figura 22. Propuesta 2 - Sistema de sujeción del soporte de los actuadores

Como tercera alternativa se tiene también una unión con tornillos, pero en vez de utilizar el perfil del antebrazo, se utilizan los travesaños de refuerzo del mismo (ver Figura 23).

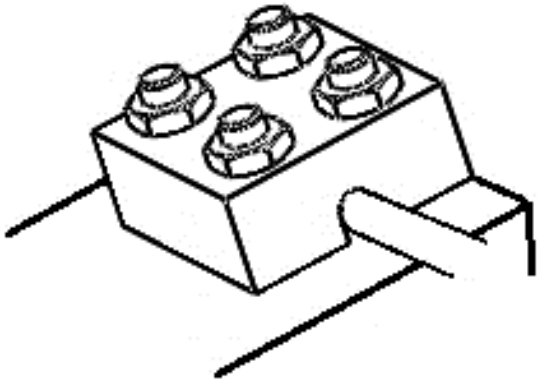


Figura 23. Propuesta 3 - Sistema de sujeción del soporte de los actuadores

3.2.2 Software para programar el control

En primer lugar, el fabricante proporciona un programa propio que permite el movimiento del brazo y el accionamiento de los distintos módulos a través de una interfaz.

Además de dicho programa, el fabricante permite diversas formas de controlar el robot. Por un lado, proporciona el protocolo de comunicación serie para poder comunicar el robot con cualquier dispositivo. Esta forma de control es la de más bajo nivel, teniendo que administrar el protocolo de comunicación y teniendo acceso únicamente a las funciones más básicas.

Por otro lado, el fabricante también proporciona una interfaz de programación de aplicaciones (API) para abstraer todas estas funciones más básicas y así permitir un control más fácil y rápido. Esta API es común para distintos lenguajes o entornos de programación. Junto con la documentación, el fabricante proporciona ejemplos para mostrar el funcionamiento de esta. En la Figura 24 se muestran las distintas capas de abstracción empleadas en los ejemplos dependiendo del lenguaje o sistema utilizado.

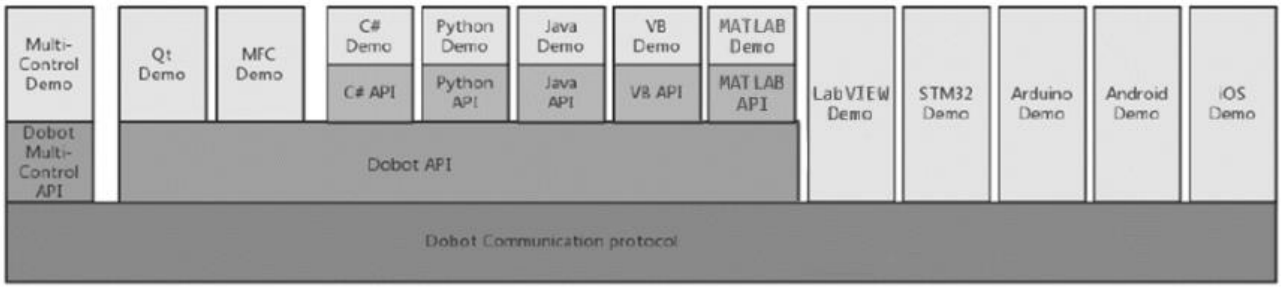


Figura 24. Posibles formas de control del brazo robot

Al permitir programar el robot desde entornos externos, pueden utilizarse herramientas o funciones existentes en estos que posibiliten un control más avanzado y fácil.

- Lenguajes de programación de alto nivel

Proporcionan una serie de instrucciones que permiten escribir secuencias de órdenes y algoritmos para controlar el comportamiento físico y lógico del robot.

Mediante el uso de lenguajes de programación de alto nivel como Java, Python, C++, etc. se puede programar cualquier tipo de aplicación o algoritmo de control. Para ello, se utilizan entornos de desarrollo que dependen del lenguaje utilizado. En la actualidad, hay disponibles una gran variedad de ellos de forma gratuita.

- *MATLAB*

MATLAB es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan la manipulación de matrices, la implementación de algoritmos y la comunicación con programas en otros lenguajes y con otros dispositivos físicos.

Además, dispone de herramientas adicionales como *Simulink* y diversas librerías que permiten ampliar las capacidades del programa.

- *Simulink*

Simulink es un entorno de simulación y diseño basado en modelos para sistemas integrados y dinámicos integrado en *MATLAB*. Básicamente es una herramienta para realizar diagramas mediante bloques gráficos con una librería personalizables de bloques.

Simulink permite incorporar algoritmos de *MATLAB* a modelos, así como exportar los resultados de las simulaciones a *MATLAB* para análisis posteriores.

- *Robotics System Toolbox*

Este conjunto de herramientas para *MATLAB* incluye algoritmos y conectividad para desarrollar aplicaciones robóticas para vehículos aéreos y terrestres, manipuladores y robots humanoides.

Para los robots manipuladores, como es este caso, incluye algoritmos de cinemática inversa, restricciones cinemáticas y dinámicas haciendo uso de una representación de solido rígido en árbol.

Las herramientas incluyen tanto funciones para *MATLAB* como bloques para utilizar con *Simulink*.

- *LabVIEW*

LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real e integrado.

3.2.3 Estrategias de control

- Control cinemático

El control cinemático puede recibir como datos de entrada el punto de destino, la precisión de posicionamiento, el tipo de trayectoria, velocidad o tiempo invertido.

Tradicionalmente, una tarea de planificación de movimiento suele tener tres tipos de resultados: rutas, caminos y trayectorias.

- Una ruta es un conjunto ordenado de configuraciones que deben ser alcanzadas por el robot.
- Por camino se entiende la discretización de una función continua que interpola las configuraciones definidas en una ruta.
- Finalmente, cuando se habla de trayectoria se está haciendo referencia a un camino que tiene asociado un perfil cinemático; es decir, a cada configuración perteneciente al camino se le asocia una velocidad.

El objetivo de la planificación de caminos consiste en determinar la secuencia de configuraciones por las que debe pasar el robot para llegar, desde la configuración inicial a la final, evitando posibles colisiones con los obstáculos del entorno de trabajo.

El problema de la planificación de la trayectoria consiste en encontrar las velocidades/aceleraciones que deben proporcionar los actuadores del sistema robótico para producir una trayectoria que evite obstáculos y lleve al sistema hasta la configuración final deseada, siendo habitual el planteamiento adicional de la minimización del tiempo de ejecución o la energía consumida.

Cuando se trata de generar trayectorias o caminos en sistemas no holónomos (con relaciones cinemáticas no lineales, como los robots de brazo), una configuración inicial y una final no pueden unirse mediante cualquier trayectoria. Las restricciones cinemáticas del sistema imponen unas condiciones que sólo algunos caminos cumplirán. En dichos casos, el problema de planificación consiste en encontrar un camino que conecte la configuración inicial con la configuración final y que, además, a lo largo del mismo se satisfagan las restricciones.

Existen múltiples métodos de planificación de movimientos, algunos de ellos son aplicables a una amplia variedad de problemas, mientras que otros tienen aplicabilidad limitada.

- Control dinámico

El robot utilizado no cuenta con una forma de medir el par realizado por los motores de las articulaciones ni tampoco proporciona datos para poder calcularlos. Por este motivo no se considera el control dinámico como una opción viable.

3.3 Selección de alternativas

3.3.1 Piezas

Atendiendo a las especificaciones establecidas en el apartado 3.1.1 se han definido variables para cada una de ellas y se les ha asignado un valor para cada alternativa.

- Unión del antebrazo con la mano

En primer lugar, se verifica que las alternativas cumplen con las especificaciones de tipo restricción. En caso de no cumplir alguna de las restricciones se descarta dicha alternativa.

Tabla 3.1. Unión del antebrazo con la mano - Restricciones

	A1	A2	A3	A4
E1	Sí	Sí	Sí	Sí
E2	Sí	Sí	Sí	Sí
E4	Sí	Sí	Sí	Sí
E8	Sí	Sí	Sí	Sí

Una vez comprobado que todas las alternativas cumplen con las restricciones, se evalúan las especificaciones restantes otorgándoles un valor numérico teniendo en cuenta la importancia de dicha especificación con respecto a las demás. Sumando estos valores se obtiene una puntuación para cada alternativa, siendo la alternativa con mayor puntuación la seleccionada.

Tabla 3.2. Unión del antebrazo con la mano - Valores de las variables asignadas a las especificaciones

	A1	A2	A3	A4
E3	5	5	5	5
E5	3.5	3.3	3.2	2.5
E6	5	5	5	5
E9	5	5	5	5
E11	0	0	0	7
Total	18.5	18.3	18.2	24.5

Como puede verse en la Tabla 3.2, todas las alternativas presentan características similares. Por otro lado, la alternativa 4 presenta una ventaja con respecto a las demás ya que permite aumentar la movilidad del robot.

- Soporte de los actuadores

En primer lugar, se verifica que las alternativas cumplen con las especificaciones de tipo restricción. En caso de no cumplir alguna de las restricciones se descarta dicha alternativa.

Tabla 3.3. Soporte de los actuadores - Restricciones

	A1	A2	A3
E1	Sí	Sí	Sí
E2	Sí	Sí	Sí
E4	Sí	Sí	Sí

E7	Sí	Sí	Sí
E8	Sí	Sí	Sí

Por tanto, se comprueba que las alternativas cumplen con todas las restricciones y se procede a asignarles valores. La alternativa seleccionada será la que tenga una mayor puntuación.

Tabla 3.4. Soporte de los actuadores - Valores de las variables asignadas a las especificaciones

	A1	**A2**	**A3**
E3	5	5	5
E5	4	5	4.5
E6	5	5	5
E9	5	5	5
E11	5	1	8
Total	24	21	27.5

En cuanto al sistema de sujeción la primera propuesta donde se hace uso de bridas para la unión se descarta por la falta de rigidez y fiabilidad de la unión.

Entre las dos alternativas restantes, la alternativa 2 necesita de menos tornillos y es más ligera que la 3, pero analizando el espacio disponible se observa que la propuesta 2 puede interferir con los mecanismos internos del brazo robot. Por lo tanto, la alternativa más viable y la que se ha seleccionado es la propuesta 3.

- Solución elegida

Elegidas las distintas propuestas de cada pieza se tiene finalmente la solución que se adopta para el diseño preliminar. En la Figura 25 se muestra el ensamblaje del brazo robot junto con las alternativas seleccionadas.

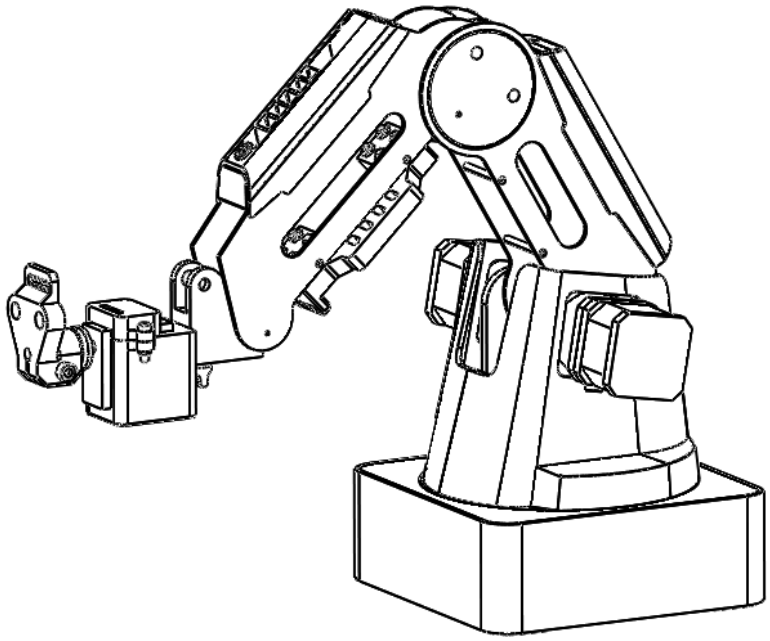


Figura 25. Primeras soluciones elegidas ensambladas en el brazo robot

3.3.2 Software de programación

Al igual que se ha hecho para la selección de las piezas, atendiendo a las especificaciones establecidas en el apartado 3.1.2 se han comprobado si las alternativas cumplen con las restricciones como muestra la Tabla 3.5.

Tabla 3.5. Restricciones del control

	Aplicación del fabricante	Lenguajes de alto nivel	MATLAB	LabVIEW
C1	Sí	Sí	Sí	Sí
C2	No	Sí	Sí	Sí
C3	Sí	Sí	Sí	Sí
C4	Sí	Sí	Sí	Sí
C5	No	Sí	Sí	Sí

La aplicación proporcionada por el fabricante queda descartada por no permitir la simulación de trayectorias ni tampoco se poderse usar en otras plataformas.

Para seleccionar las alternativas restantes se le asignan valores a cada especificación como se muestra en la Tabla 3.6.

Tabla 3.6. Restricciones del control

	Lenguajes de alto nivel	MATLAB	LabVIEW
C6	7	5	2
C7	2	6	6
Total	9	11	8

Teniendo en cuenta que se trata de un proyecto donde el coste económico es de gran importancia y que éste dependerá tanto del coste directo del software como del tiempo empleado, se concluye que *MATLAB* es la mejor solución en este caso. Los lenguajes de programación a pesar de ser gratuitos en su mayoría requieren de mucho más tiempo para desarrollar los programas lo que acaba incrementando el coste total.

4 DISEÑO PRELIMINAR

En primera instancia, no se consideraba la posibilidad de disponer de giro en la muñeca. Tras contemplar esta opción, se han buscado alternativas que permitan dicho giro y finalmente se ha conseguido añadir este grado de libertad al movimiento del brazo robot.

Por otro lado, se ha replanteado la selección del sistema de sujeción, puesto que, la solución elegida no era la más adecuada.

En un principio se desconocían algunos detalles de control por lo que se consideró la posibilidad de añadir un soporte adicional para un posible controlador (por ejemplo, una placa Arduino). Tras analizar con más detalle el robot y la documentación del fabricante se observó que todo el control puede realizarse desde el ordenador, por lo que se ha descartado el soporte adicional mencionado.

4.1 Diseño de las piezas

Tras el proceso de selección de alternativas realizado durante la fase conceptual del trabajo y tras las modificaciones y correcciones realizadas, se ha realizado un primer diseño de todas las piezas necesarias para acoplar las manos protésicas al brazo robot.

Para diseñar estas piezas se ha considerado en primera instancia que el proceso de fabricación que se utilizará será el de impresión 3D. Por este motivo, algunas de las geometrías que se muestran a continuación están condicionadas por las geometrías posibles mediante este proceso de fabricación. Además, se han añadido algunas características para facilitar la fabricación.

4.1.1 Acople de la mano

Esta pieza será la responsable de unir la mano con el servomotor que acciona el giro de la muñeca. Consta de tres partes diferenciadas. En la parte superior de la Figura 26 se puede ver los agujeros donde irán situados los tubos que servirán de guía para los tendones, en la parte central se encuentran los agujeros para los tornillos que unen la mano y finalmente en la parte inferior aparece el sistema tipo brida para acoplar la pieza al servomotor.

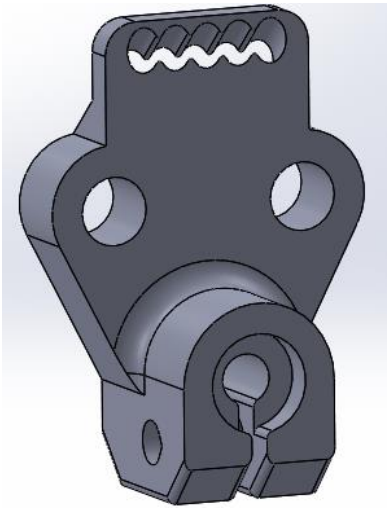


Figura 26. Acoplador de la mano

Los tubos se unen a la pieza a presión o bien con algún tipo de adhesivo en caso de ser necesario.

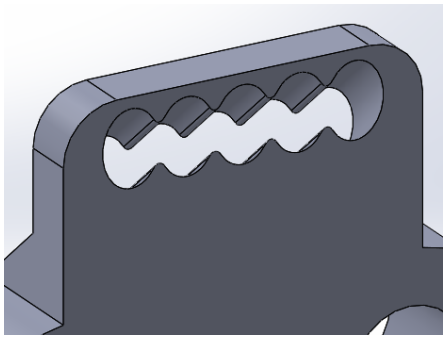


Figura 27. Acoplador de la mano. Detalle del soporte de los tubos guía

En la Figura 28 se puede ver la unión entre el acoplador de la mano y el servomotor. Esta unión se realiza por fricción al apretar el tornillo de la unión tipo brida.

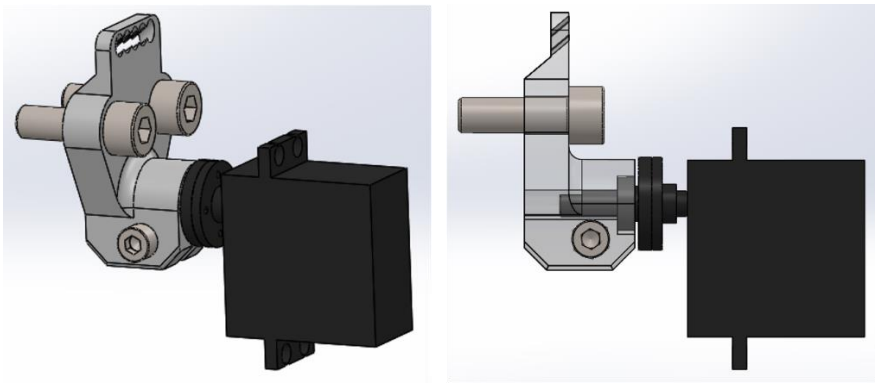


Figura 28. Ensamblaje del acoplador de la mano

4.1.2 Soporte del servo

La función de esta pieza es la de sujetar el servomotor que acciona la articulación de la muñeca. El propio robot cuenta con una pieza para realizar esta sujeción, pero la orientación final del servomotor no es adecuada para los estudios posteriores que se quieren realizar. Se aprovechará por tanto el sistema de acople existente para el acople de la nueva pieza. El soporte consta de dos piezas como puede verse en la Figura 29.

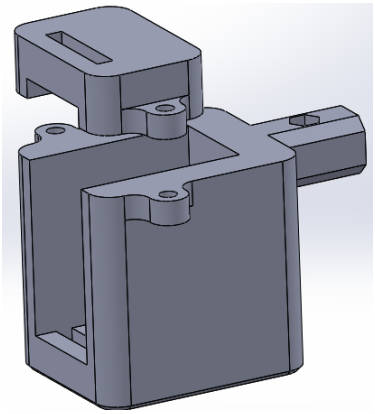


Figura 29. Soporte del servo

Para el acople de la pieza al brazo robot se utilizará un sistema similar al utilizado en el robot para acoplar los sus distintos módulos. Puesto que se planea utilizar materiales ligeros, los cuales tienen por lo general una menor resistencia y dureza, se ha modificado ligeramente la forma original de realizar el acople. En vez de presionar el extremo de la palometa contra la pieza, se utilizará una tuerca que repartirá la presión sobre una mayor superficie. (ver Figura 30 y Figura 31)

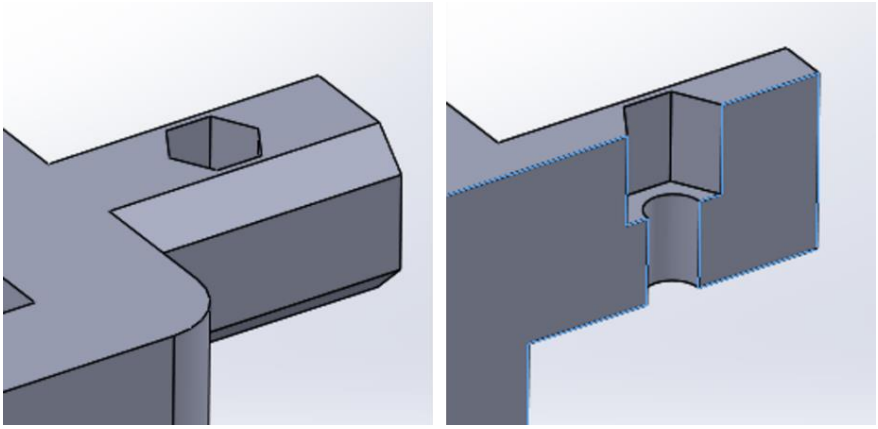


Figura 30. Soporte del servo. Detalle del acople con el brazo robot

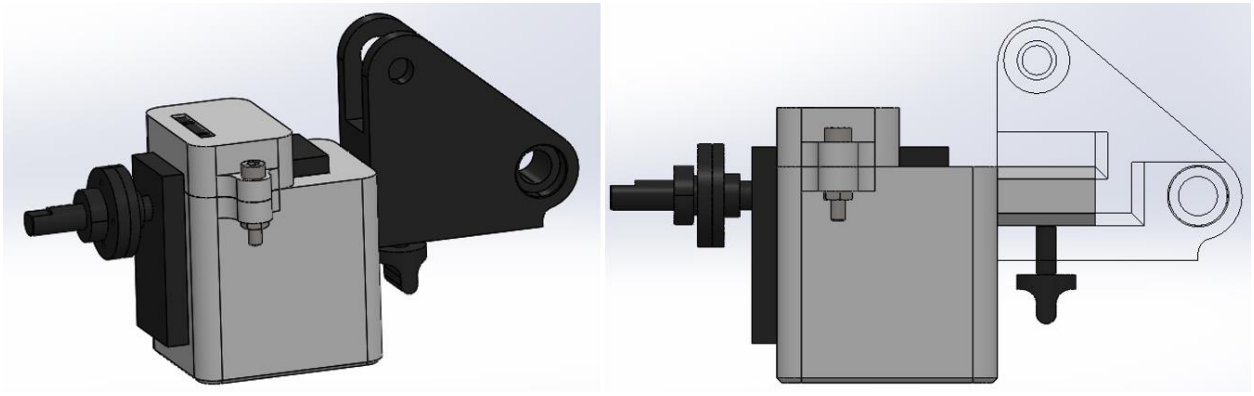


Figura 31. Ensamblaje del soporte del servo

La sujeción del servo se obtiene por la geometría del soporte uniendo las dos piezas de este mediante tornillos.

4.1.3 Soporte de los actuadores

Esta pieza servirá únicamente en el caso de acoplar una mano protésica basada en tendones. Su función consiste en soportar los distintos actuadores que moverán los tendones que accionan los dedos. Los actuadores se acoplan mediante tornillos haciendo uso de las ranuras. Por tanto, para acoplar cada tipo o tamaño de actuador será necesaria otra pieza de unión adicional.

Se sitúa en la parte del antebrazo del robot y se acopla al mismo haciendo uso de los refuerzos transversales existentes. Según se muestra en la Figura 32 el soporte consta de una pieza principal donde irán situados los actuadores. Las piezas restantes servirán para la sujeción del soporte al brazo robot.

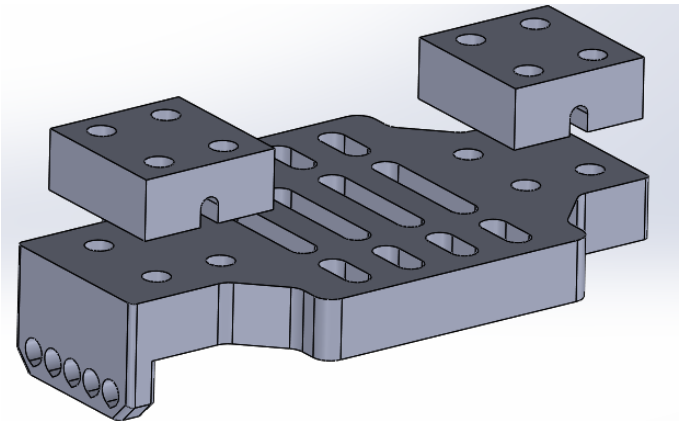


Figura 32. Soporte de los actuadores

Al igual que en el acople de la mano, se ha diseñado en la pieza una sujeción para los tubos que servirán de guía para los tendones (ver Figura 33).

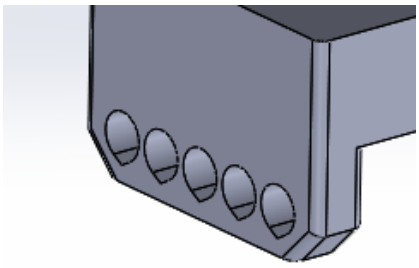


Figura 33. Soporte de los actuadores. Detalle de la sujeción de los tubos guía

4.1.4 Ensamblaje en el brazo robot

En la Figura 34 se muestran todas las piezas ensambladas en el brazo robot, además también aparecen los tubos guía para los tendones.

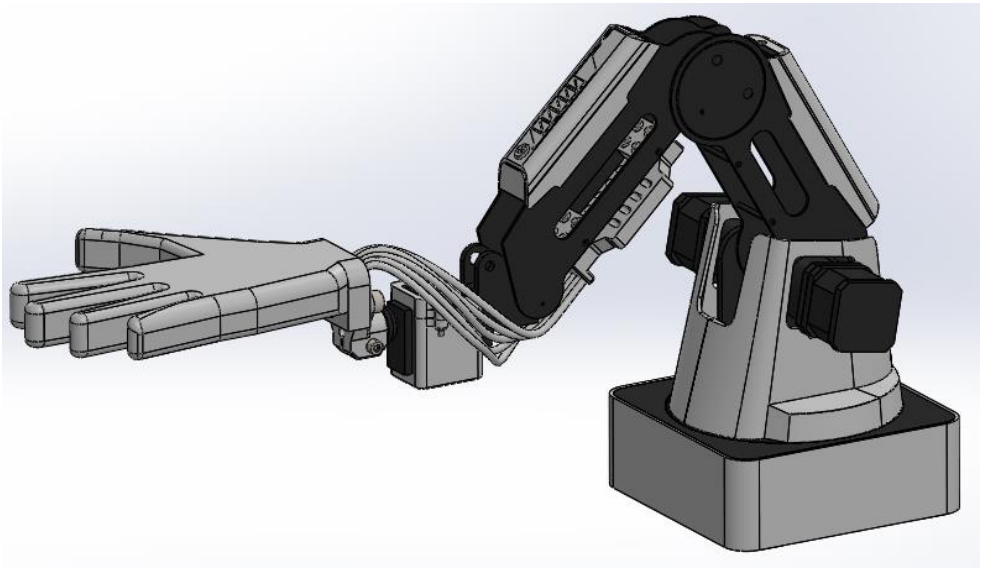


Figura 34. Piezas ensambladas en el brazo robot.

5 REDISEÑO DE LAS PIEZAS

Tras realizar el diseño preliminar de todas las piezas se han realizado varias modificaciones. Estas modificaciones han sido motivadas por los resultados de los análisis de resistencia y por necesidades de diseño.

5.1 Soporte del servo

El diseño preliminar de las dos piezas del soporte del servo presenta problemas de montaje. Para que se pueda ensamblar correctamente se han realizado modificaciones a ambas piezas.

Las modificaciones que se han realizado son debidas al eje del servo. En un principio se consideraba que estaba unido rígidamente al servo, pero al desmontar las piezas se observó que el eje del servo era una pieza independiente del servo y no se encontraba restringida axialmente por el mismo. La restricción axial venía dada por el soporte metálico, por lo que al eliminarlo del nuevo montaje se perdía esta restricción.

La parte superior del soporte del servo (ver Figura 29) se ha modificado para añadir la restricción axial al eje del servo. Además, con esta modificación se consigue reducir el momento soportado por el servo, mejorando así la durabilidad y robustez del montaje (ver Figura 35).

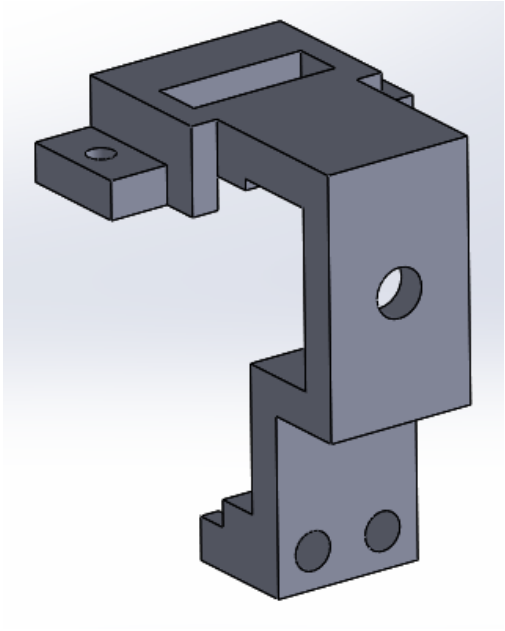


Figura 35. Modificación de la pieza superior del soporte del servo

Se han realizado varias modificaciones a la pieza inferior del soporte del servo para dejar hueco a los tornillos que servirán para sujetar la otra pieza según se muestra en la Figura 36.

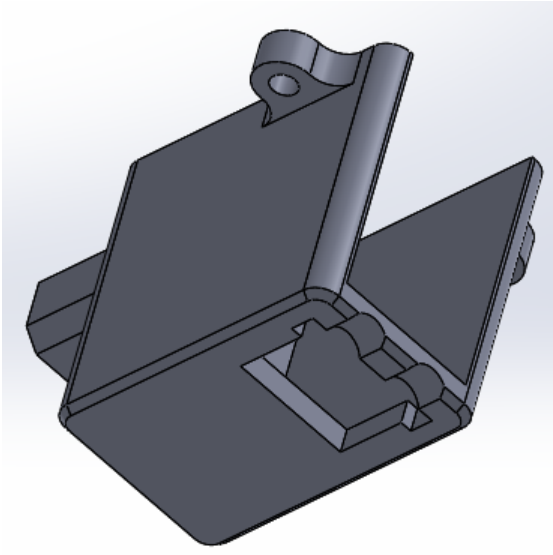


Figura 36. Modificación de la pieza inferior del soporte del servo

Finalmente, puede verse el ensamblaje de estas nuevas piezas junto con el servo en la Figura 37.

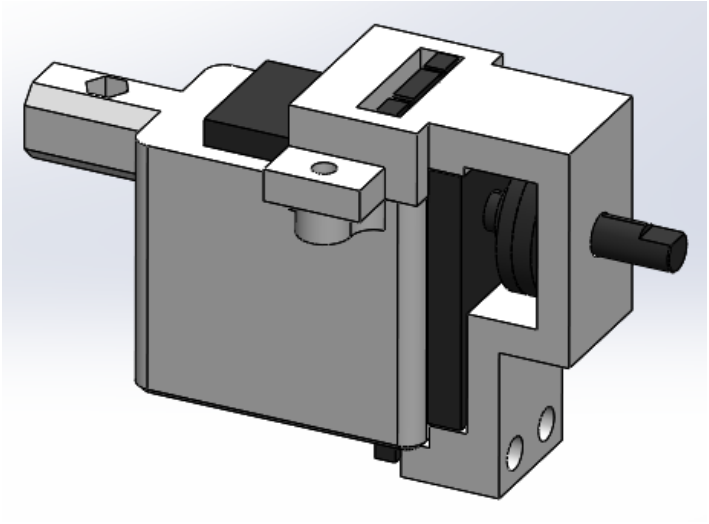


Figura 37. Ensamblaje del soporte del servo después de las modificaciones

6 DISEÑO DE DETALLE

6.1 Diseño final de las piezas

Tras seleccionar entre las alternativas planteadas en el diseño conceptual y tras las modificaciones realizadas en el diseño preliminar se ha obtenido el diseño final de todas las piezas.

Una vez seleccionado el diseño de las piezas se han dimensionado para que cumplan con los requisitos de resistencia y rigidez. Finalmente, las piezas obtenidas pueden verse en la Figura 38 montadas en el brazo robot.

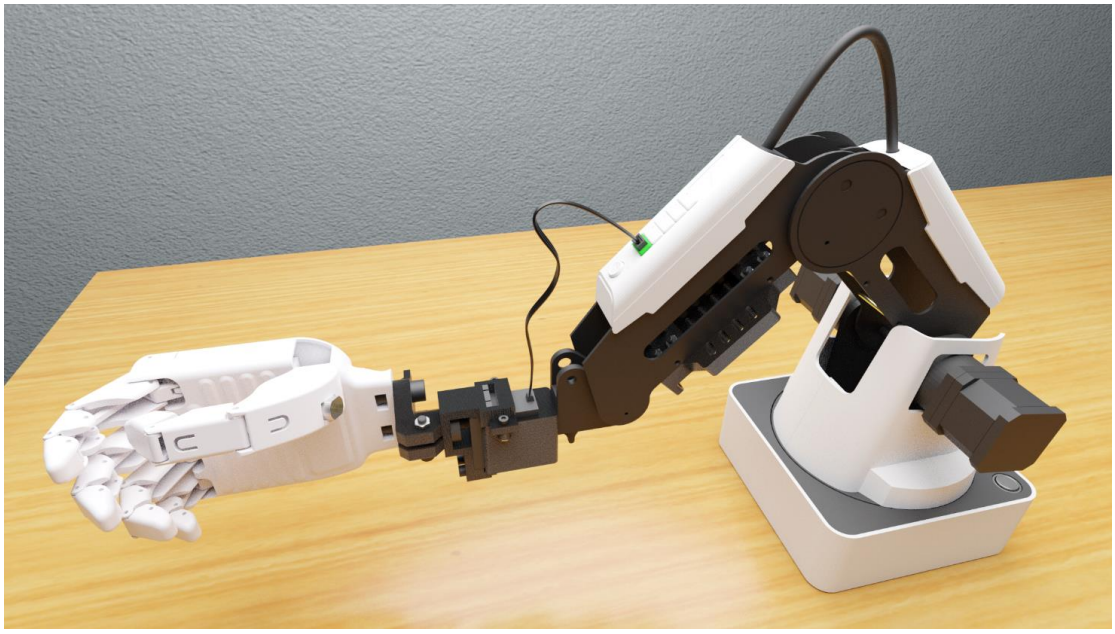


Figura 38. Ensamblaje del diseño final

Las piezas diseñadas pueden dividir en tres grupos:

- El soporte del servo. Se compone de dos piezas y sirve para alojar el servo y acoplarlo al brazo robot.

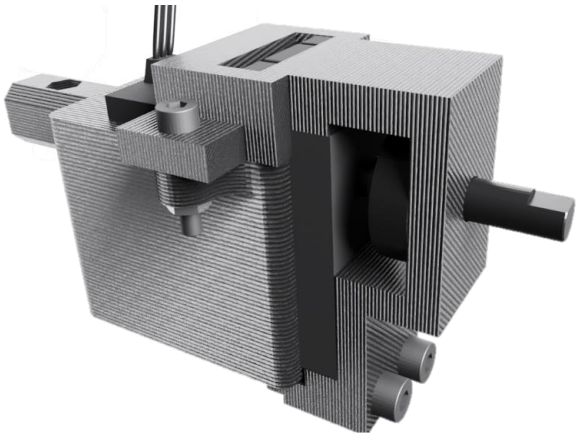


Figura 39. Soporte del servo

- El acoplador de la mano. Es una única pieza que sirve para unir la mano protésica al eje del servo.

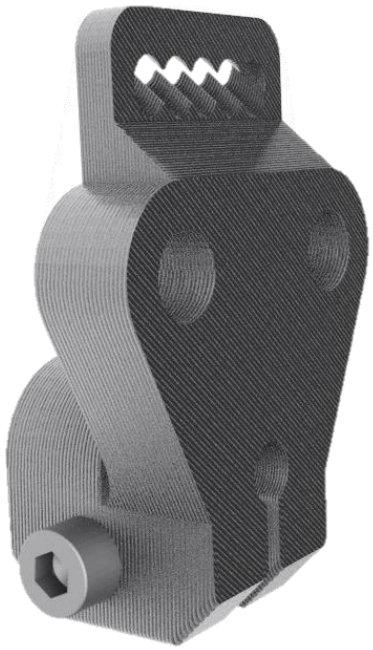


Figura 40. Acoplador de la mano

- El soporte de los actuadores. Se compone de dos piezas distintas, la pieza base y la pieza que sirve para anclar la pieza base al robot. Su función es incorporar puntos de anclaje para los actuadores que tiren de los tendones en el caso de utilizar una mano protésica basada en tendones.

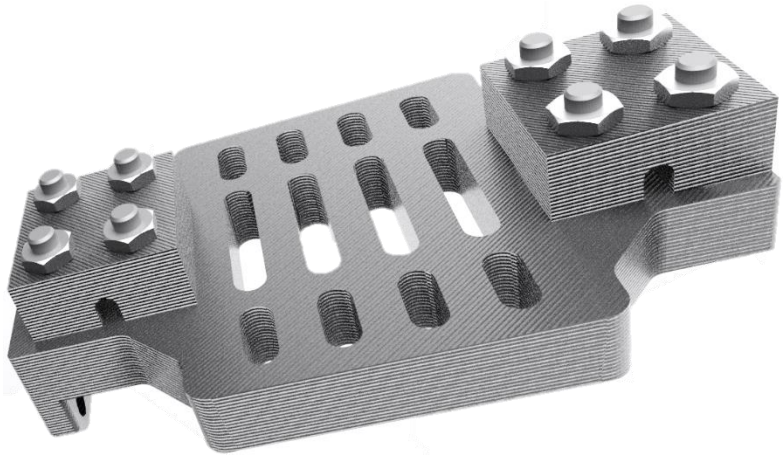


Figura 41. Soporte de los actuadores

Finalmente, en la Figura 42 se puede ver el robot situado en el entorno de trabajo. En esta figura pueden apreciarse las escalas del robot y las piezas fabricadas.

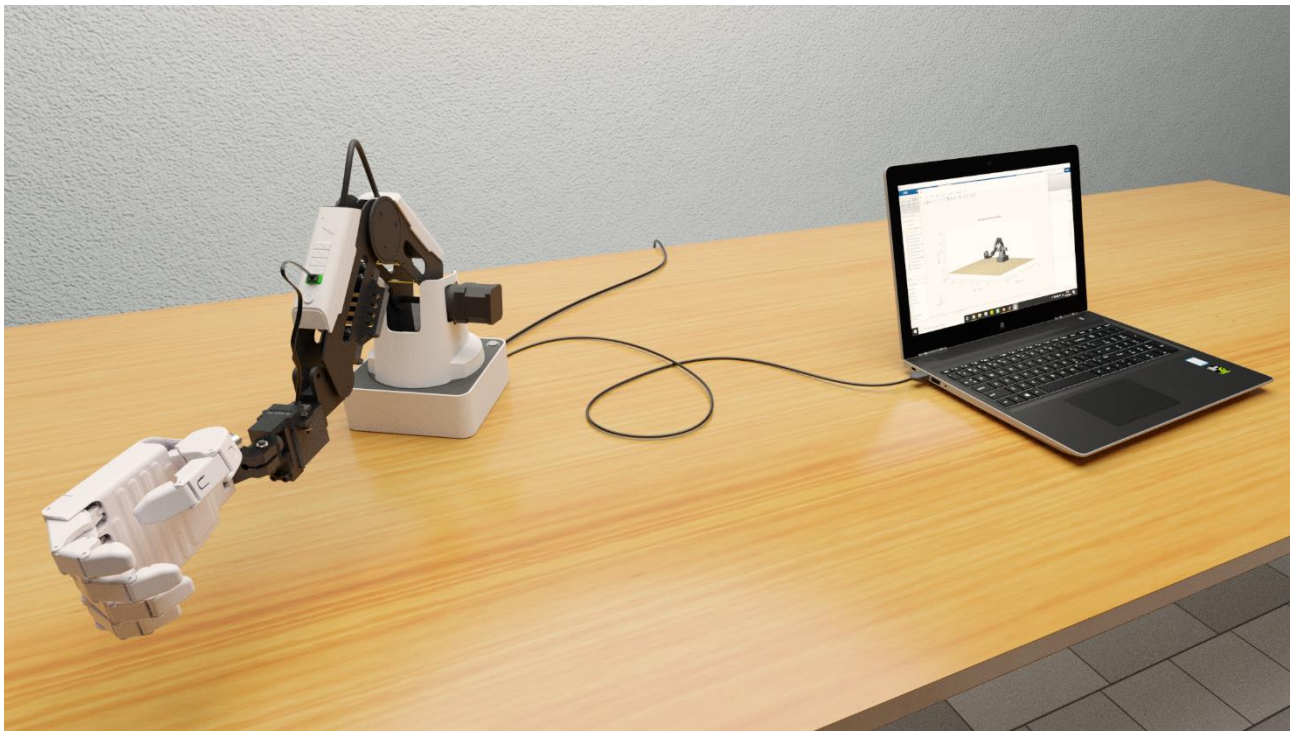


Figura 42. Brazo robot modificado en el entorno de trabajo

6.2 Modificación del brazo robot con las piezas fabricadas

A continuación, se detalla el proceso de desmontaje y montaje necesarios para la modificación.

Puesto que se pretende reutilizar el servo incluido en uno de los módulos (ver Figura 43) proporcionados junto con el robot, primero será necesario desmontar dicho módulo.

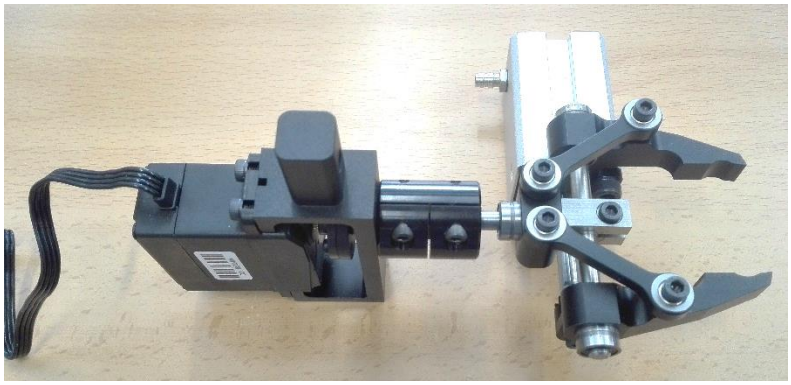


Figura 43. Servo en el módulo de la pinza

Para desmontar el servo del ensamblaje es necesario desatornillar el servo del soporte metálico y aflojar el acoplador que une el eje del servo con la pinza (ver Figura 44).

Figura 44. Módulo de la pinza desmontado

Una vez desmontado el servo se pasa a realizar el ensamblaje del sistema de acoplamiento diseñado.

En primer lugar, se coloca el eje del servo en el puente del servo como se muestra en la Figura 45.

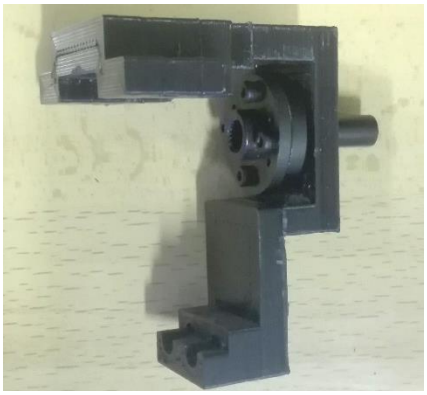


Figura 45. Ensamblaje del eje del servo y del soporte del servo

Por otro lado, se ensamblan el servo y el soporte del servo según se muestra en la Figura 46.

Figura 46. Ensamblaje del servo y el soporte del servo

Una vez se tienen estos dos subensamblajes se pueden unir mediante tornillos según se muestra en la Figura 47.

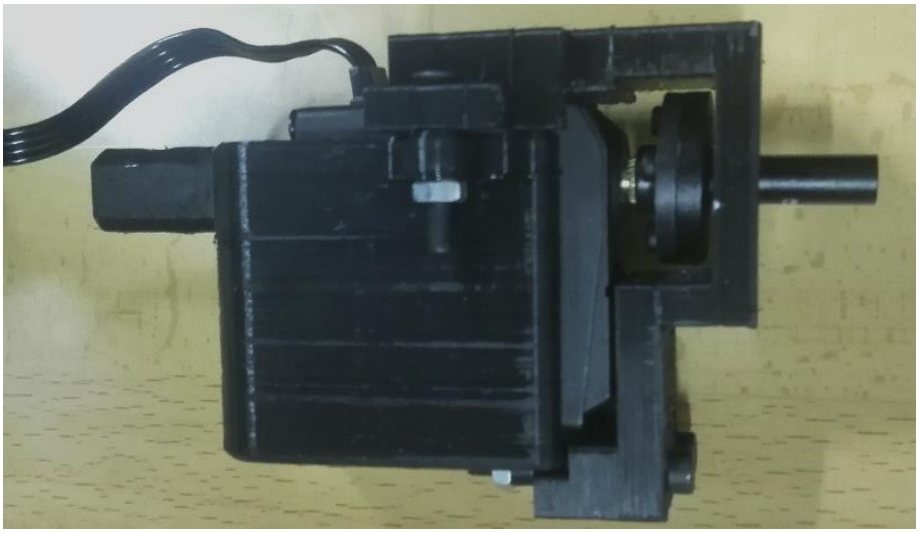


Figura 47. Unión de los dos subensamblajes

Tras ensamblar este conjunto, se puede incorporar al brazo robot y conectar el cable del servo según se muestra en la Figura 48.

Figura 48. Acople del soporte del servo al brazo robot

Antes de montar el acoplador de la mano sobre el eje del servo se atornilla a la mano protésica que se quiera montar (ver Figura 49).

Figura 49. Acoplador unido a la mano

Por último, se acopla este ensamblaje al eje del servo (ver Figura 50).

Figura 50. Mano acoplada al eje del servo

En la Figura 51 puede verse como queda el conjunto tras el ensamblaje.

Figura 51. Mano acoplada al brazo robot

6.3 Análisis de resistencia de las piezas

Se han realizado estudios estáticos basados en elementos finitos con dos objetivos. Por un lado, para optimizar el diseño de las piezas con el fin de que tengan únicamente el material necesario para satisfacer los criterios de diseño y, por otro lado, para asegurar que las tensiones que aparecerán en las piezas durante funcionamiento del robot no superan el límite elástico del material (PLA en este caso).

El proceso de optimización ha consistido en la variación del espesor de pared de las piezas. Se ha decidido variar este parámetro ya que el software para la fabricación por impresión 3D permite producir las piezas huecas indicando un espesor de pared partiendo del modelo macizo. Esto ahorra tiempo de diseño puesto que no es necesario modificar el modelo de partida.

Dependiendo de la pieza analizada se han utilizado criterios distintos. En los casos en los que la deformación de la pieza influye significativamente al desplazamiento del efector final, se ha priorizado la rigidez, aumentando el espesor de pared hasta obtener desplazamientos razonables. En los casos en los que las deformaciones son de menor importancia, se ha ajustado el valor del espesor de la pared hasta obtener un factor de seguridad entre 1,5 y 2,5.

Como resultado se ha obtenido el espesor de pared del acoplador de la mano (3,2 mm), el puente del servo (maciza) y el soporte del servo (2,8 mm).
Para ver más detalles sobre los análisis realizados, consultar el

Anexo I: Análisis de resistencia de las piezas.

6.4 Requisitos funcionales de las piezas y tolerancias

Los agujeros de las piezas utilizados para el ensamblaje mediante tornillos deben cumplir unas tolerancias tales que aseguren que se podrán unir las piezas una vez se fabriquen.

Para ello los agujeros deberán cumplir la siguiente condición:

$$H \geq F + T$$

*Donde H es el agujero en su condición de máximo material, F es el valor del perno en su condición virtual (en este caso se ha tomado el valor nominal del diámetro al tratarse de elementos comerciales) y T es la tolerancia de posición del agujero.

Para los agujeros de los tornillos se ha utilizado una tolerancia E10 por lo que se puede obtener la tolerancia de posición T mediante despejando de la condición anterior.

$$T \leq H - F$$

De igual forma para asegurar que los ajustes entre piezas que encajan entre ellas son los deseados, será necesario establecer las tolerancias adecuadas.

Considerando que la calidad superficial obtenida es basta y tomando el sistema de eje único, se han tomado ajustes h11/H11 para las piezas que deban encajar entre ellas sin que haya mucho juego entre ellas y ajustes h11/E10 en los casos en que se requiere movimiento relativo entre las piezas encajadas.

Para consultar las tolerancias finales asignadas, consultar el documento PLANOS.

6.4.1 Cadena de cotas

Se ha analizado la desviación que podrá sufrir el efector final en los ejes Y y Z a causa de las tolerancias de fabricación y del montaje. Esta desviación supone un cambio en las coordenadas del efector final. La desviación en el eje Y puede ser fácilmente compensada en el software cambiando la coordenada h del efector final (Ver X). En cuanto a la deviación en el eje X, sería más complejo de compensar en el software ya que requeriría la modificación de las ecuaciones de la cinemática.

Para calcular la desviación se ha estudiado una cadena de cotas siguiendo el orden de piezas mostrado en la Figura 52 utilizando el complemento *TolAnalyst* de *SolidWorks*. El cálculo se ha realizado sin tener en cuenta las tolerancias de la mano acoplada, ya que pueden variar notablemente dependiendo de la mano. Para obtener las medidas de la desviación del efector final ha sido necesario crear una pieza auxiliar (pieza 6), la cual se ha acotado con cotas exactas para que no influya en el resultado.

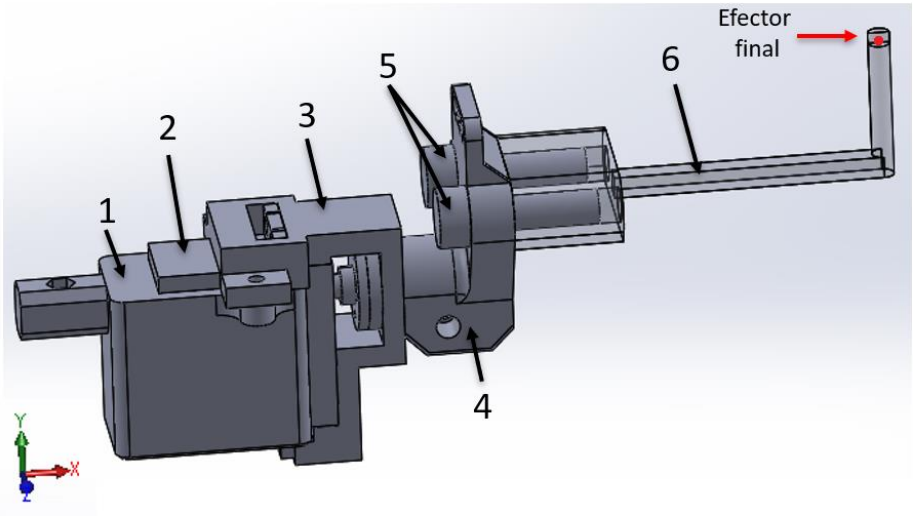


Figura 52. Orden de la cadena de cotas

La desviación producida por las cadenas de cotas se ha medido desde el plano medio del saliente que permite la unión con el resto del brazo hasta el punto definido para el efector final (ver Figura 53)

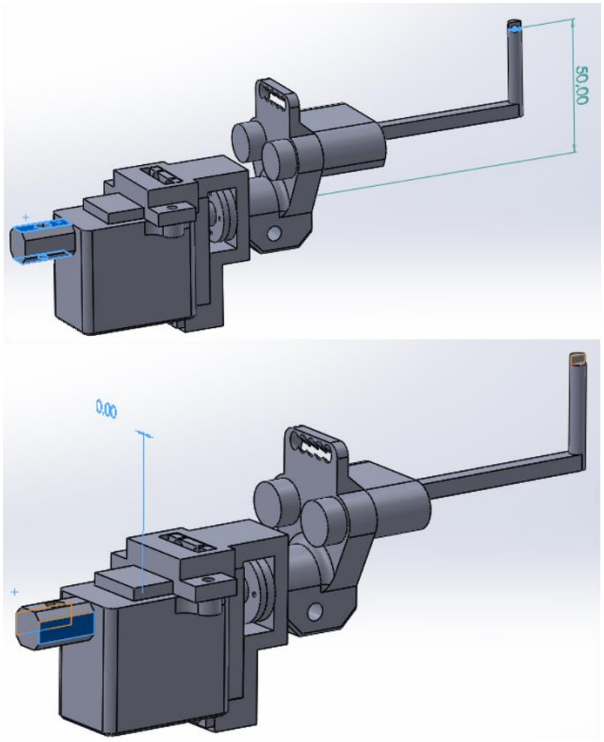


Figura 53. Desviaciones medidas en las cadenas de cotas. En el eje Z (Arriba) y en el eje Y (Abajo)

Para la obtención de los resultados se han tenido en cuenta las tolerancias de orientación y se han flotado los pasadores. Tras los primeros resultados obtenidos, se comprobó la desviación obtenida y la contribución de cada cota o tolerancia a la desviación total. En función de la contribución se fueron modificando las tolerancias hasta obtener una desviación inferior a $\pm 0.5\ mm$.

Como resultado final se han obtenido los valores mostrados en la Figura 54.

Figura 54. Resultados de las cadenas de cotas

La mayor contribución a la desviación en ambos ejes ha sido por parte de los tornillos (pieza 5). Esto se debe a que los agujeros del acoplador de la mano (pieza 4) son de diámetro ligeramente superior al de los tornillos para asegurar un fácil montaje. Una posible solución si se quiere reducir estas desviaciones sería disminuir el diámetro de los agujeros, aunque podría resultar en un ensamblaje forzado de los tornillos. Otra solución para disminuir el diámetro de los agujeros sería hacerlos roscados, lo que podría facilitar el montaje. Las contribuciones finales de cada cota proporcionadas por *SolidWorks* son las que se muestran en la Figura 55.

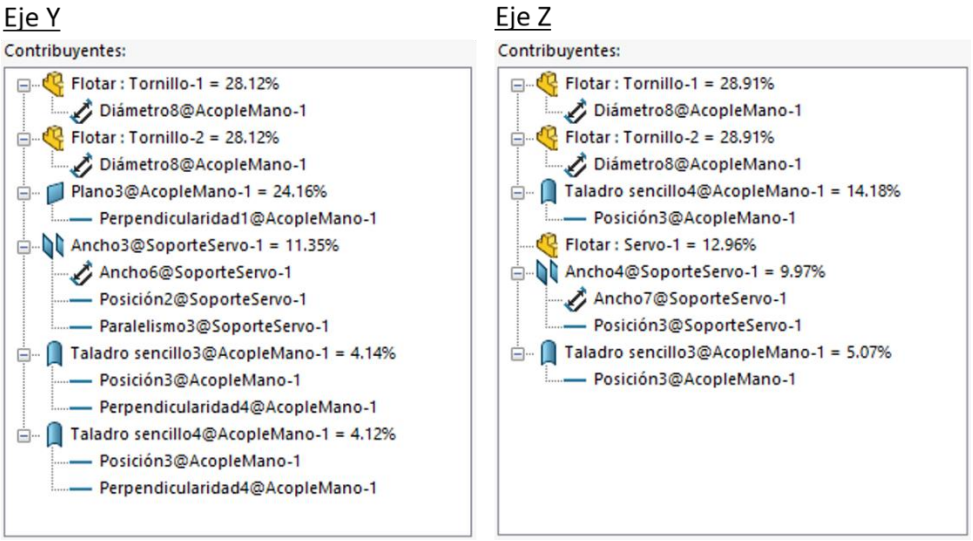


Figura 55. Contribuciones a la desviación en la cadena de cotas

6.5 Seguridad

Aunque el robot utilizado tiene marcado CE y por lo tanto debería cumplir con la normativa europea en cuanto a seguridad, se han identificado los peligros y se han evaluado los riesgos que aparecerán durante su uso. También se comprobará que la modificación realizada no incorpora nuevos peligros. En la Figura 56 puede verse la placa informativa del robot donde aparece el marcado CE junto con el marcado de otras normativas, características eléctricas del robot e información sobre la empresa.

Figura 56. Etiqueta informativa en la parte inferior del robot

6.5.1 Normativa aplicable

En primer lugar, se han identificado la normativa aplicable y los peligros que contempla cada normativa.

Peligro	Normativa aplicable
No garantizar la seguridad de la máquina	Real Decreto 1644/2008, por el que se establecen las normas para la comercialización y puesta en servicio de las máquinas.
Contacto eléctrico directo o indirecto	Real Decreto 187/2016, por el que se regulan las exigencias de seguridad del material eléctrico destinado a ser utilizado en determinados límites de tensión.
Compatibilidad eléctrica y electrónica de los equipos	Real Decreto 186/2016, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos.
Puesta en marcha mientras haya personas en zonas peligrosas	UNE-EN ISO 14118:2018. Seguridad de las máquinas. Prevención de puesta en marcha intempestiva.
Peligro asociado a no poseer un sistema de parada de emergencia para cualquier tipo de energía utilizada.	UNE-EN ISO 13850:20116. Seguridad de las máquinas. Función de parada de emergencia. Principios de diseño.
Peligro asociado a no posicionar los protectores con respecto a la velocidad de aproximación de partes del cuerpo humano, determinando las distancias mínimas entre la zona de detección de órganos de accionamiento de los protectores y la zona peligrosa.	UNE-EN ISO 13855:2011. Seguridad de las máquinas. Posicionamiento de los protectores con respecto a la velocidad de aproximación a partes del cuerpo.
Peligro asociado a una mala evaluación y verificación de los riesgos.	UNE-EN ISO 12100:2012. Seguridad de las máquinas. Principios generales para el diseño. Evaluación del riesgo y reducción del riesgo.
Peligro asociado a la fase de montaje, transporte y mantenimiento.	

6.5.2 Evaluación de riesgos

El riesgo relativo a un peligro se estima en función de la gravedad del daño que pueda producir y de la probabilidad de que ocurra. La probabilidad, a su vez, depende del tiempo de exposición de la persona al peligro, de las veces que ocurre el suceso peligroso y de la probabilidad de evitar el daño o limitarlo.

Las escalas utilizadas para valorar la probabilidad, las consecuencias y el riesgo se muestran en Tabla 6.1, Tabla 6.2 y Tabla 6.3.

Tabla 6.1. Escala para la probabilidad (Basada en una jornada de 8 horas)

Baja	Podría darse al menos 1 vez cada 10 años
Media	Podría darse al menos 1 vez al año
Alta	Podría darse al menos 1 vez al mes
Muy alta	Podría darse al menos 1 vez por semana

Tabla 6.2. Escala para las consecuencias

Leve	Pequeñas lesiones que no requieren hospitalización
Moderada	Lesiones con incapacidad laboral transitoria
Grave	Lesiones graves irreparables
Muy grave	Muerte

Tabla 6.3. Escala para los riesgos

>70	Inasumible	Requiere actuación inmediata preferentemente mediante prevención intrínseca y si no es posible mediante protección
35-70	Alto	Requiere actuación al menos mediante medidas de protección
10-35	Intermedio	Deben estudiarse alternativas para intentar reducir el riesgo. Uso de EPIs. Avisos. Inclusión en manual de instrucciones
<10	Bajo	Informar del riesgo en manual de instrucciones. EPIs

Analizando cada peligro independientemente se obtiene la Tabla 6.4, donde se calcula el nivel de riesgo para ver si es necesario realizar alguna acción para reducirlo.

Tabla 6.4. Evaluación de los riesgos

Evento que genera el riesgo	Tipo de peligro	Fase	Probabilidad	Consecuencias	Riesgo (0-100)	Valoración del riesgo
1. Contusión en alguna parte del cuerpo debido a partes agudas del robot	Mecánico	Montaje, uso, mantenimiento, limpieza, retirada	Media	Leves	10	Bajo
2. Impacto debido al movimiento del robot	Mecánico	Uso	Muy alta	Leves	20	Intermedio

3. Enganche entre las partes móviles del robot	Mecánico	Uso	Alta	Leves	16	Intermedio
4. Electrocución por partes activas	Eléctrico	Uso, mantenimiento	Baja	Leves	4	Bajo
5. Quemaduras por cortocircuito	Eléctrico	Uso, mantenimiento	Baja	Moderadas	10	Bajo
6. Quemaduras por elementos a altas temperaturas al fabricar las piezas	Térmico	Fabricación	Media	Leves	10	Bajo

El riesgo obtenido para todos los peligros identificados es intermedio o bajo. En los casos en los que el riesgo es bajo se ha comprobado que se informa adecuadamente de los mismos tanto en el caso del robot como en el de la impresora 3D en sus respectivos manuales de instrucciones. Para los peligros de riesgo intermedio, dado que no se pueden reducir debido a que son intrínsecos al uso que quiere darse al robot, se comprueba que el robot cuenta con los avisos necesarios y que también se informa de los peligros y se indican las precauciones a tener en cuenta en el manual de instrucciones.

En la Figura 57 puede verse la señalización del peligro de atrapamiento.

Figura 57. Aviso de peligro de atrapamiento

El icono utilizado para señalizar el peligro no se corresponde con la normativa vigente (UNE-EN ISO 7010:2012/A7:2017). Por este motivo sería conveniente modificar dicho aviso con el símbolo indicado por la normativa (ver Figura 58). Adicionalmente, sería recomendable orientar el aviso de

modo que el símbolo y el texto se vean correctamente, puesto que en la orientación actual el aviso queda boca abajo durante el uso normal del robot.

Figura 58. Señalización de riesgo de atrapamiento según ISO 7010:2012/A7:2017

En cuanto a la fuente de alimentación eléctrica tiene su propio marcado CE, ya que es fabricada por otra empresa distinta y cuenta con una carcasa de plástico adecuada que evita el riesgo de contacto eléctrico directo e indirecto. En la Figura 59 se puede ver el marcado CE junto con la información del fabricante y características de la fuente de alimentación.

Figura 59. Etiqueta informativa en la fuente de alimentación

6.6 Fabricación

6.6.1 Selección del proceso de fabricación y del material

En cuanto a la fabricación de las piezas se ha optado por la impresión 3D por varios motivos. El primero es que permite fabricar las piezas a un coste bajo en comparación a procesos basados en la mecanización o la inyección. Además, el propio robot dispone de un módulo de impresión 3D por lo que en un principio existe la posibilidad de que el robot imprima sus propias piezas. Otra característica interesante de este proceso de fabricación es que permite fabricar las piezas con un interior hueco o casi hueco, lo que permite la creación de piezas muy ligeras.

En cuanto al material utilizado los materiales más comúnmente utilizados son:

- Ácido poliláctico (PLA): Es el más utilizado de todos por su facilidad de uso. En comparación al resto de materiales es más rígido, aunque también más frágil.

- Acrilonitrilo butadieno estireno (ABS): También muy utilizado. Mejor resistencia a los golpes que el PLA y más barato por lo general. El principal inconveniente es la mayor dificultad a la hora de fabricar las piezas al necesitar de una envolvente para mantener la pieza caliente durante su impresión.

- Tereftalato de polietileno modificado con glicol (PETG): Un poco menos común. Resistencia comparable a la del ABS. El precio es ligeramente superior al del PLA. En cuanto a facilidad de impresión, es más sencillo que el ABS, aunque algo más complejo que el PLA.

También existen otros termoplásticos que se pueden utilizar en el proceso de impresión 3D como plásticos con fibras, nylon, elastómeros, etc. Estos materiales normalmente requieren de condiciones especiales para poder utilizarse y además su precio es notablemente superior. Por otro lado, no existen especificaciones que hagan necesario el uso de alguno de estos materiales por lo que se descartan como alternativas viables.

Analizando las ventajas y desventajas de cada uno de los termoplásticos se decide elegir el PLA por ser el más sencillo de utilizar y por su mayor rigidez.

6.6.2 Proceso de fabricación

La selección del proceso de fabricación ha condicionado algunas características del diseño de las piezas.

Todas las piezas han sido diseñadas de forma que sea posible fabricarlas mediante impresión 3D. Por otro lado, se han tenido en cuenta las tolerancias típicas obtenidas en este tipo de proceso de fabricación para asegurar un correcto ensamblaje posterior. Por último, se han realizado algunas modificaciones para facilitar la impresión.

En primer lugar, se han exportado todos los modelos CAD generados en *SolidWorks* a formato STL. Este es el formato utilizado por el software para la fabricación por impresión 3D. Se he optado por una calidad alta del mallado de triángulos para que las piezas resultantes tengan una geometría lo más semejante posible al modelo CAD. En la Figura 60 se muestra un ejemplo de la malla obtenida para el acoplador de la mano.

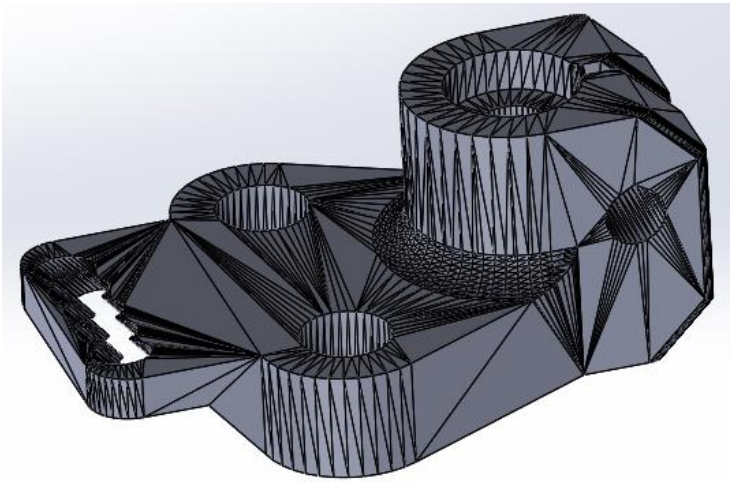


Figura 60. Mallado del acoplador de la mano

Una vez exportados los archivos de todas las piezas a un formato se define la orientación en la que se imprimirá cada pieza. Este es un parámetro crítico que condiciona la fabricabilidad y la calidad de las piezas.

- ## Acoplador de la mano

En la Figura 61. Orientación de impresión del acoplador de la mano Figura 61 se muestra la orientación en la que se debe fabricar la pieza.

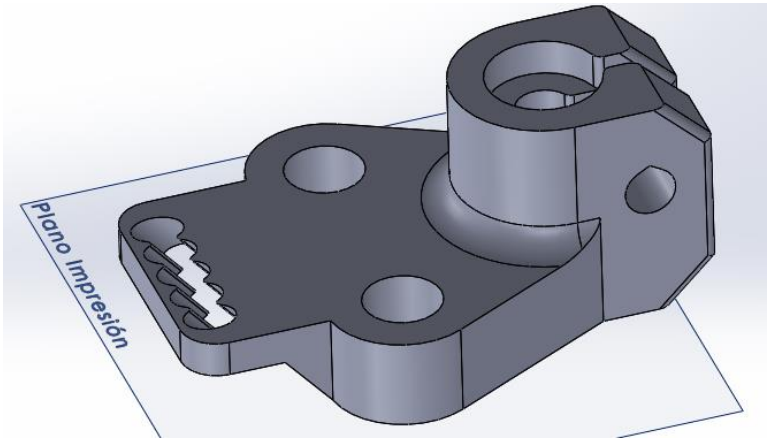


Figura 61. Orientación de impresión del acoplador de la mano

Como puede observarse existe una zona que queda en voladizo y que por lo tanto no puede imprimirse sin la ayuda de soportes. En la Figura 62 se muestra la zona donde deben colocarse dichos soportes.

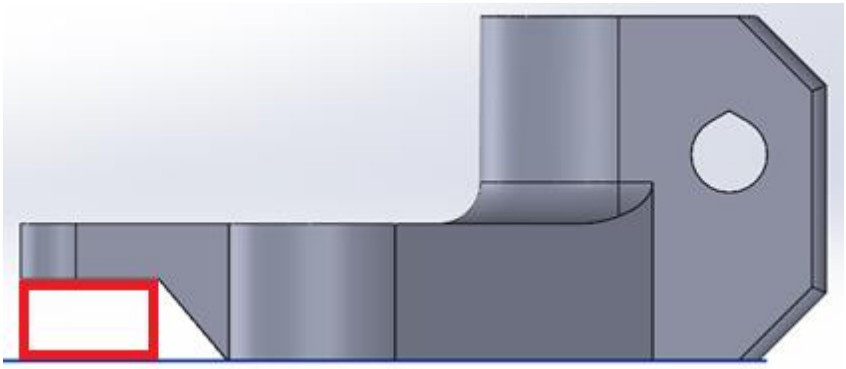


Figura 62. Soporte necesario para fabricar el acoplador de la mano

En la parte superior de los agujeros no perpendiculares a la base de impresión el material depositado durante la fabricación dispone de muy poco soporte por lo que fluye ligeramente, esto provoca que el diámetro resultante sea inferior al de diseño. Para ello se ha realizado la modificación que se observa en la Figura 63 para contrarrestar este efecto y asegurar que el tornillo que se insertará en dicho agujero encaja sin problemas.

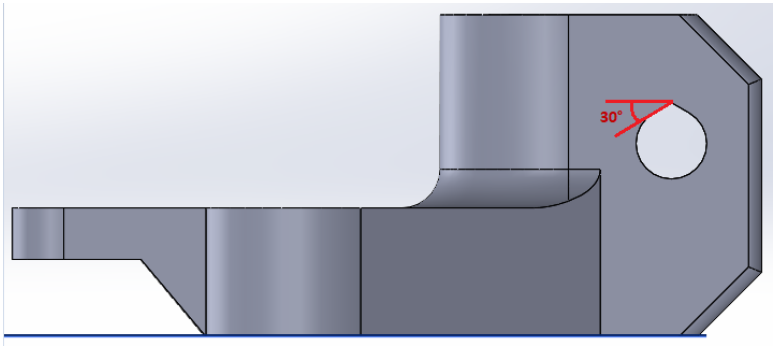


Figura 63. Modificación de agujero en el acoplador de la mano para facilitar la impresión

- ## Soporte del servo

El soporte del servo se compone de dos partes, la primera sirve para acoplar el soporte al brazo robot y la segunda para sujetar el servo.

En la Figura 64 puede observarse la orientación en la que se debe imprimir la primera parte del soporte.

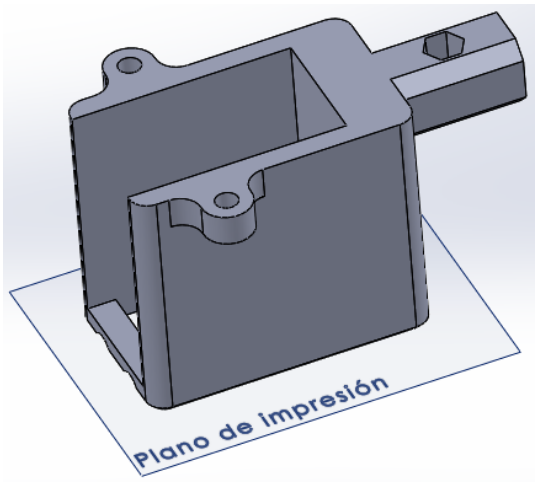


Figura 64. Orientación de impresión de la primera parte del soporte del servo

Puesto que con la orientación elegida aparecen zonas que se imprimirán en el aire al fabricarlas, es necesario añadir soportes para posibilitar la fabricación (zona roja en la Figura 65).

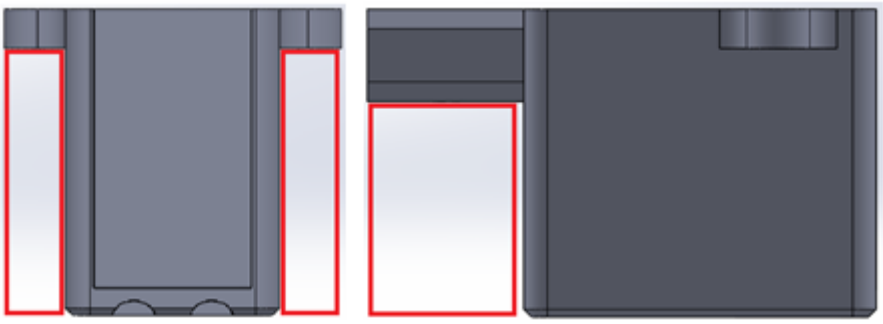


Figura 65. Soportes necesarios para la fabricación del soporte del servo

En cuanto a la segunda parte del soporte del servo se orienta según muestra la Figura 66.

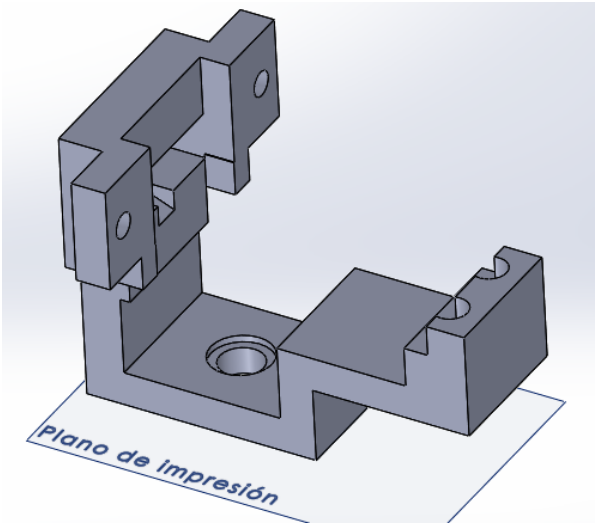


Figura 66. Orientación de la parte 2 del soporte del servo

En esta pieza existen varias zonas donde se requiere de soportes para la correcta fabricación de la pieza como se muestra en la Figura 67.

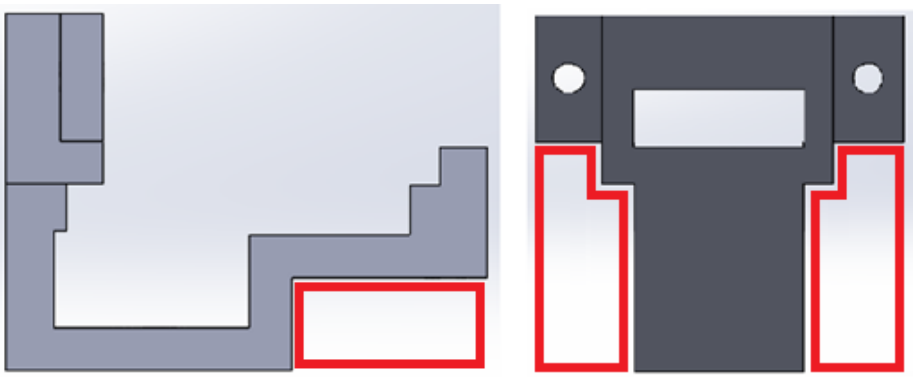


Figura 67. Soportes necesarios de la parte 2 del soporte del servo

- ## Soporte de los actuadores

La orientación de fabricación de la base del soporte de los actuadores se muestra en la Figura 68.

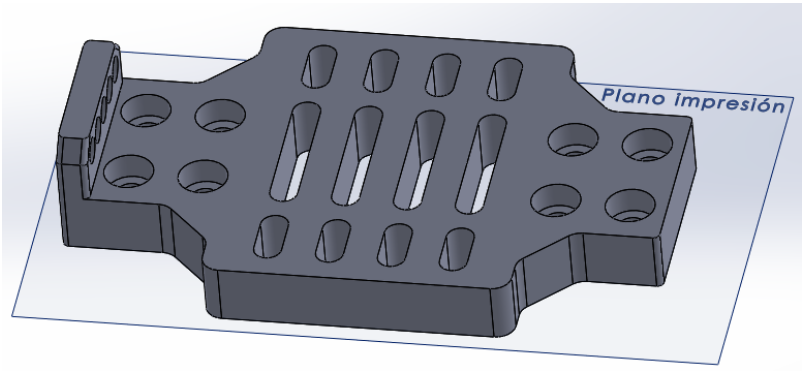


Figura 68. Orientación de la base del soporte para los actuadores

Los agujeros no perpendiculares a la base suelen resultar con dimensiones más pequeñas. Por lo que se ha añadido la misma modificación a los agujeros donde irán colocados los tubos guía.

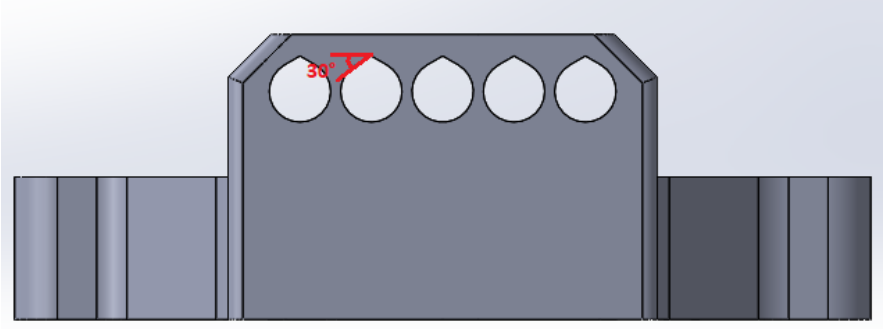


Figura 69. Modificación de agujeros en la base del soporte parara los actuadores para facilitar la impresión

Finalmente, en la Figura 70 se muestra la orientación para fabricar el acoplador del soporte de los actuadores.

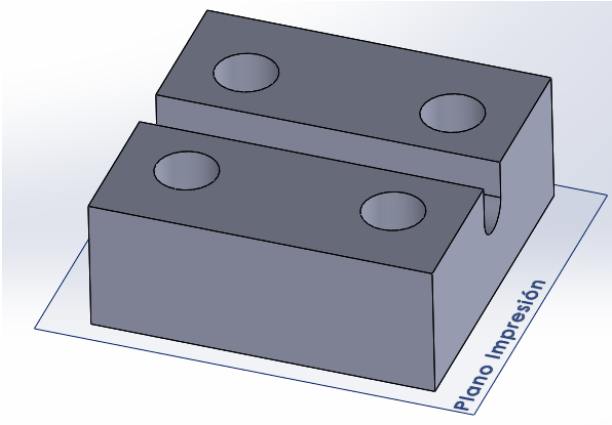


Figura 70. Orientación del acoplador del soporte de los actuadores

6.7 Sistema de control

Para realizar el sistema de control se ha utilizado *MATLAB*. Además de las funcionalidades básicas que proporciona el programa, se ha hecho uso de las herramientas en *Robotics System Toolbox.* Para poder comunicar *MATLAB* con el brazo robot se ha cargado la librería proporcionada por el fabricante.

Los pasos llevados a cabo para realizar la programación del control han sido los siguientes:

- Modelado del robot: En primer lugar, se ha definido la estructura del robot dentro del programa. Esto incluye los sistemas de referencia locales, los tipos de enlace y sus posiciones límite y el modelo geométrico.
- Cálculo de la cinemática inversa: A partir de un punto del efector final se calcula el ángulo de cada unión.
- Elaboración de una API propia: Para facilitar el uso de la API proporcionada por el fabricante y evitar errores.
- Control del robot: A partir de los cálculos de cinemática inversa se envían los comandos necesarios para mover el robot.

Se ha decidido utilizar el inglés para la programación para facilitar su uso por el mayor número de persona posible.

6.7.1 Modelado del sistema en MATLAB

Para representar la estructura de los robots *Robotics System Toolbox* utiliza un árbol de sólidos rígidos. El árbol está compuesto de sólidos rígidos conectados mediante uniones. Cada sólido rígido tiene una unión que define como se mueve dicho cuerpo con respecto a su padre en el árbol. La posición de un cuerpo con respecto al siguiente se define mediante una transformación homogénea.

- Árbol de solidos rígidos

Base: todo árbol contiene una base. La base define las coordenadas globales y es el primer punto de anclaje para un sólido rígido.

Sólido rígido: Es el componente básico para construir el árbol. Cuando se añade un sólido a un árbol es asociado mediante relaciones padre-hijo con otros sólidos de dicho árbol. Cada sólido rígido tiene además un sistema de coordenadas asociado y una única unión (unión con el sólido padre). El sistema de coordenadas del sólido rígido y de la unión son coincidentes.

Unión: Cada sólido rígido contiene una unión que define su movimiento respecto a su padre. Los tipos de unión soportados son de revolución, fija y prismática.

Para cada unión se define un eje de movimiento. El eje de la unión es un vector de 3 dimensiones que define el eje de rotación o la dirección de traslación dependiendo de si se trata de una articulación o de un par prismático.

Cada unión también contiene una posición de reposo y límites de movimiento.

Por otro lado, en cada unión se aplica una transformación homogénea entre el sistema de coordenadas del sólido hijo y del sólido padre.

Utilizando estas estructuras se ha modelado el brazo robot dentro de *MATLAB*. Se han tomado algunas consideraciones en cuenta. Por ejemplo, todos los sistemas de referencia de los sólidos rígidos se han definido de forma que el eje Z coincida con el eje de rotación de la articulación. La dirección de dicho eje Z y él eje X también se ha establecido para que coincida con los definidos en el brazo robot por el fabricante en la medida de lo posible. Además, se ha añadido un sólido rígido adicional unido rígidamente a la mano (efector final) que servirá para posicionar la mano cuando se calcule la cinemática inversa. En la Figura 71 se muestra la estructura obtenida en una configuración del robot arbitraria.

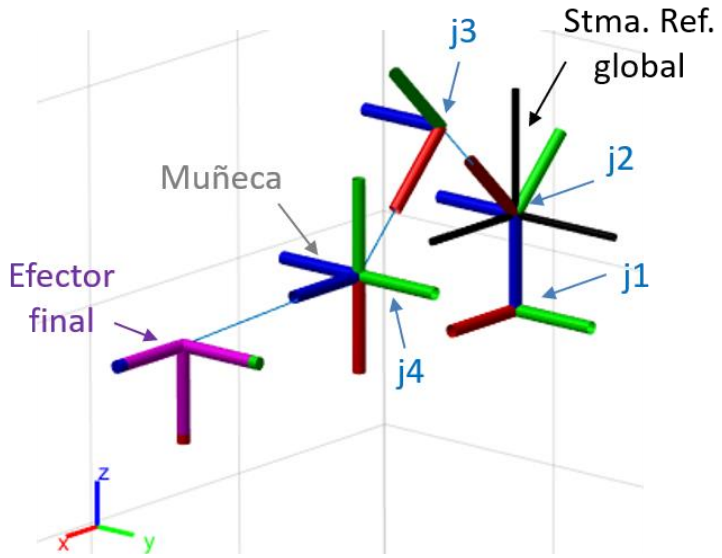


Figura 71. Estructura del modelo en MATLAB del brazo robot

Una vez definida esta estructura, *MATLAB* cuenta con toda la información necesaria sobre la geometría y cinemática del robot. No obstante, para facilitar la visualización y comprensión de los resultados que se obtengan en el futuro se han añadido las mallas obtenidas de los modelos CAD de cada parte del robot, asignándolas a su correspondiente sólido rígido. Tras realizar este paso se puede ver el modelo del robot dentro de *MATLAB* como se muestra en la Figura 72.

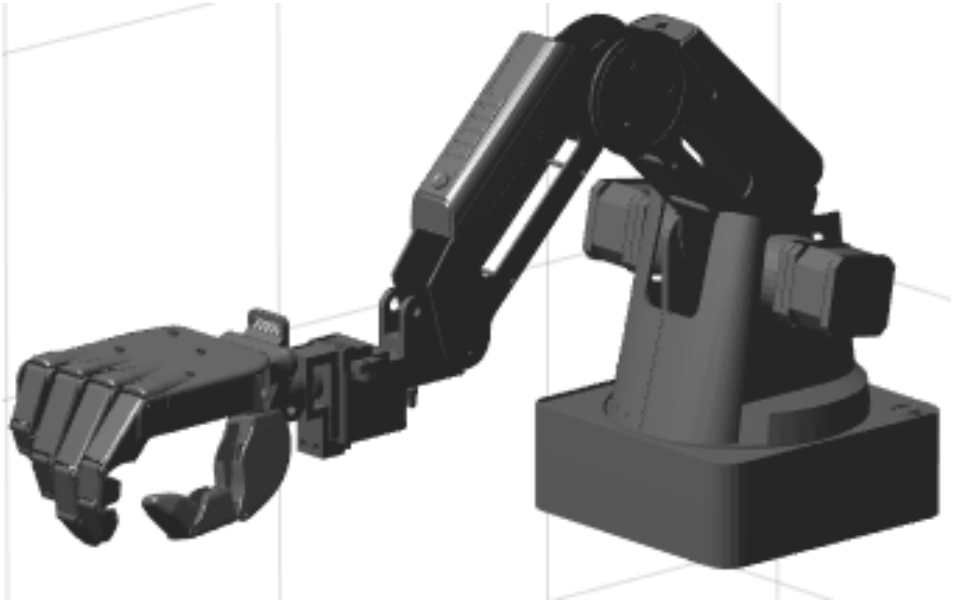


Figura 72. Mallas aplicadas a los sólidos rígidos en MATLAB

Todo el código utilizado para la definición del modelo del robot en *MATLAB* puede consultarse en el Anexo IV: Código de MATLAB.

6.7.2 Cálculo de la cinemática directa

Para obtener el modelo cinemático directo del brazo robot se ha utilizado el algoritmo de Denavit-Hartenberg [11].

Primero, se han definido los sistemas de referencia de cada eslabón y se han identificado los distintos elementos siguiendo el método, según se muestra en la Figura 73.

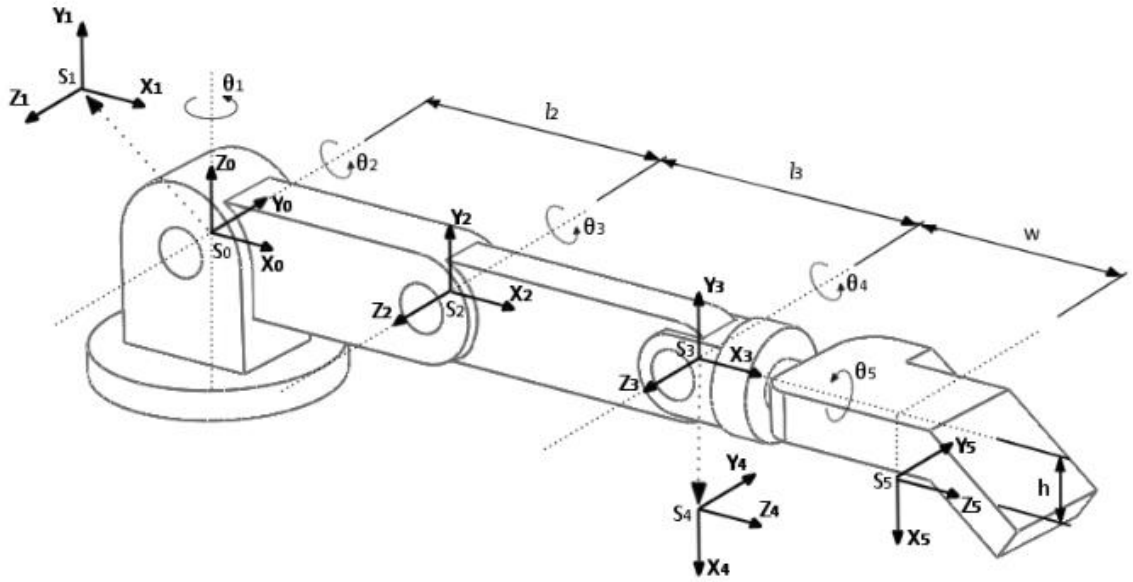


Figura 73. Definición de sistemas de referencia e identificación según Denavit-Hartenberg

Siguiendo el procedimiento marcado por el método de Denavit-Hartenberg se obtiene la matriz de transformación homogénea T.

La matriz de transformación T relaciona la posición y orientación del efector final del robot con respecto a la base en funcion de las coordenadas articulares $(\theta_1,\ \theta_2,\ \theta_3,\ \theta_5)$ según la ecuación (1)(29). El ángulo θ_4 está determinado por el mecanismo interno del brazo y depende de los ángulos θ_2 y θ_3 según la ecuación (2).

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(1)}$$

$$\theta_4 = -(\theta_2 + \theta_3) \quad \textbf{(2)}$$

La terna ortonormal de vectores n, o, a representa la orientación y el vector p representa la posición.
Para más más información sobre los cálculos realizados para la obtención de los términos de la matriz T consultar

Anexo III: Cálculo de la cinemática.

Mediante la matriz de transformación T se pueden obtener las coordenadas (r_x, r_y, r_z) de un vector r en el sistema de coordenadas del eslabón base a partir de sus coordenadas (r_u, r_v, r_w) en el sistema de coordenadas del efector final (ecuación (3)).

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} r_u \\ r_v \\ r_w \\ 1 \end{bmatrix} \quad \textbf{(3)}$$

Por ejemplo, para obtener la posición del efector final bastaría con sustituir (r_u, r_v, r_w) por el vector nulo según se muestra a continuación.

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

6.7.3 Cálculo de la cinemática inversa

Para obtener la cinemática inversa se ha realizado un desacoplo cinemático entre la parte del brazo y la de la muñeca y la mano. De este modo, partiendo de la posición deseada del efector final (p_x, p_y, p_z) y del ángulo θ_5 de la muñeca, se ha obtenido la posición $(p'_x,\ p'_y,\ p'_z)$ del sistema de referencia S_4. Finalmente utilizando estas nuevas coordenadas, se obtiene la cinemática inversa de la parte del brazo.
Las expresiones corresponden a la solución del problema cinemático inverso del brazo robot. Para consultar cómo se han obtenido estas expresiones consultar

Anexo III: Cálculo de la cinemática.

$$\theta_1 = \arctan\left(\frac{p_y}{p_x}\right) - \operatorname{arcsen}\left(\frac{h \cdot sen\ \theta_5}{\sqrt{p_x^2 + p_y^2}}\right) \quad \textbf{(4)}$$

$$\theta_2 = \arctan\left(\frac{p'_z}{\sqrt{{p'_x}^2 + {p'_y}^2}}\right) + \arctan\left(\frac{l_3 \cdot sen\ \theta_3}{l_2 + l_3 \cdot \cos\theta_3}\right) \quad \textbf{(5)}$$

$$\theta_3 = \arccos\left(\frac{p_x'^2 + p_y'^2 + p_z'^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3}\right) \quad \textbf{(6)}$$

No obstante, el fabricante utiliza un sistema de referencia distinto para definir los ángulos de las distintas articulaciones. Estos ángulos se obtienen a partir de los que se resultan de la cinemática inversa ($\theta_1, \theta_2\ y\ \theta_3$) según las ecuaciones (7), (8) y (9).

$$j_1 = \theta_1 \quad \textbf{(7)}$$

$$j_2 = 90° - \theta_2 \quad \textbf{(8)}$$

$$j_3 = -(\theta_2 + \theta_3) \quad \textbf{(9)}$$

6.7.4 Cálculo de velocidades

Además de conocer la relación entre las coordenadas articulares y las del efector final, es interesante disponer de la relación entre sus velocidades para poder conseguir que el extremo desarrolle una trayectoria temporal concreta. Esta relación en el caso del robot estudiado se puede obtener a través de la matriz jacobiana J según la ecuación (56).

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = J \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} \quad \textbf{(10)}$$

Siendo la matriz jacobiana:

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} & \frac{\partial p_x}{\partial q_5} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} & \frac{\partial p_y}{\partial q_5} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} & \frac{\partial p_z}{\partial q_5} \end{bmatrix}$$

Para obtener las velocidades articulares a partir de las velocidades del efector final es necesario obtener la matriz jacobiana inversa. Dado que en este caso no es invertible por no ser una matriz cuadrada, se obtiene su matriz pseudoinversa J^+ [12] según la ecuación (11).

$$J^+ = (J^t \cdot J)^{-1} \cdot J^t \quad \textbf{(11)}$$

Y finalmente se pueden calcular las velocidades articulares según la ecuación (12).

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} = J^+ \cdot \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} \quad \textbf{(12)}$$

6.7.5 Elaboración de API propia

La librería proporcionada por el fabricante se basa en la importación de una librería dinámica (dll) en *MATLAB*. La forma de utilizar este tipo de librerías desde *MATLAB* es mediante llamadas a la librería a través de otra función (ver Figura 74). El uso directo de estas funciones durante la escritura del código aumenta el tiempo necesario para elaborar programas más complejos y hace que sea más fácil cometer errores. Además, para obtener información del uso de cada función es necesario recurrir continuamente a la documentación del fabricante mientras que elaborando una API propia se pueden añadir comentarios y aclaraciones dentro del propio *MATLAB*.

```
calllib('DobotDll','ConnectDobot',addr_cptr, baudrate);
```

Figura 74. Llamada a la función ConnectDobot de la librería DobotDll

Las funciones o clases creadas para la nueva API se han nombrado igual que las de la librería proporcionada por el fabricante, añadiendo el prefijo 'Mg' (**Mag**ician) para denotar que se trata de la librería propia para el control del robot. Además, añadiendo dicho prefijo se evita el conflicto con otras posibles funciones con el mismo nombre.

En cuanto a la funcionalidad se ha procurado mantener la de la API original del fabricante por lo que la documentación proporcionada por el mismo es igualmente válida para la nueva API.

A continuación, se explican todas las funciones que se han elaborado para la API.

- **MgLoadDll**

Carga la librería dinámica proporcionada por el fabricante para poder acceder a sus funciones desde *MATLAB*. Junto con *MgUnloadDll* es la única funcion que no pertenece a la librería dinámica.

Esta función no recibe ningún dato de entrada ni proporciona ninguna variable de salida. En caso de que la librería ya se encuentre cargada aparecerá un mensaje indicándolo en la ventana de comandos de *MATLAB*.

- **MgUnloadDll**

Libera la memoria asignada a librería del fabricante. Junto con *MgLoadDll* es la única funcion que no pertenece a la librería dinámica.

Esta función no recibe ningún dato de entrada ni proporciona ninguna variable de salida. En caso de que la librería no se encuentre cargada aparecerá un mensaje indicándolo en la ventana de comandos de *MATLAB*.

- **MgSearchDobot**

Busca el dispositivo *Dobot* conectado al PC y devuelve su dirección en caso de encontrarlo. Esta dirección será una variable de tipo cadena de caracteres con el nombre del puerto COM al que está conectado el robot (Por ejemplo: 'COM 3'). En caso de no encontrar el dispositivo, la variable de salida será una cadena de caracteres vacía y se mostrará un mensaje en la ventana de comandos de *MATLAB* indicando el error.

Esta función no recibe ningún dato de entrada.

- **MgConnectDobot**

Establece la conexión con el brazo robot conectado en el puerto COM indicado en la variable de entrada. Esta variable de entrada es de tipo cadena de caracteres. (ver función *MgSearchDobot*).

Esta función solo funcionará correctamente si el robot no se encuentra ya conectado.

Como salida devuelve una variable de tipo entero que indica el resultado de la conexión:

0: Conexión establecida correctamente.

1: Error de conexión.

2: Tiempo de conexión agotado.

En todos los casos se muestra un mensaje en la ventana de comandos con el resultado.

- **MgDisconnectDobot**

Desconecta el dispositivo *Dobot*. Esta función no recibe ningún parámetro de entrada ni devuelve ninguna salida.

Esta función solo funcionará correctamente si el robot se encuentra conectado (ver *MgConnectDobot*).

- **MgSetPTPCmd**

Añade un comando punto a punto (PTP) a la cola de comandos del robot. Como parámetro de entrada toma una variable de tipo *MgPTPCmd*.

Devuelve una variable de tipo entero indicando la posición del comando dentro de la cola de comandos.

- **MgPTPCmd**

En este caso no se trata de una función sino de un tipo de variable o clase. Para crear una variable u objeto de este tipo es necesario especificar el modo de funcionamiento del robot (ver apartado 2.1.2) como un parámetro de tipo entero (ver *MgPTPMode*) y 4 parámetros de tipo coma flotante (x, y, z, r) que representan la información de las coordenadas objetivo del comando.

La interpretación de la información de las coordenadas dependerá del modo de funcionamiento especificado.

- **MgPTPMode**

Se trata de una enumeración. A cada elemento contenido en esta enumeración se le asigna un valor entero correspondiente a un modo de funcionamiento PTP. El objetivo es usar estos elementos (los cuales tienen un nombre que facilita recordar a qué modo se refieren) en vez de utilizar directamente números enteros. Los parámetros de la enumeración con sus respectivos valores se muestran a continuación:

JUMP_XYZ = 0
MOVJ_XYZ = 1
MOVL_XYZ = 2
JUMP_ANGLE = 3
MOVJ_ANGLE = 4
MOVL_ANGLE = 5
MOVJ_INC = 6
MOVL_INC = 7
MOVJ_XYZ_INC = 8
JUMP_MOVL_XYZ = 9

- **MgSetQueuedCmdStartExec**

Esta función da comienzo a la ejecución de los comandos añadidos a la cola. No tiene ningún parámetro de entrada o salida.

- **MgSetQueuedCmdStopExec**

Detiene la ejecución de los comandos en la cola. Esta función no fuerza la parada del comando que se encuentre en ejecución en el momento de la llamada a la función. No tiene ningún parámetro de entrada o salida.

- **MgGetQueuedCmdCurrentIndex**

Devuelve un entero indicando la posición del comando en ejecución dentro de la cola. Esta función es útil para determinar cuándo se han ejecutado todos o una parte de los comandos. No tiene ningún parámetro de entrada.

- **MgSetWAITCmd**

Añade un comando de espera a la cola de comandos. Como entrada tiene una variable de tipo entero en milisegundos. El comando de espera hace que el robot permanezca quieto durante el tiempo especificado antes de continuar ejecutando el resto de los comandos en la cola.

Como salida devuelve un entero que indica la posición del comando de espera en la cola.

Todo el código correspondiente a este apartado puede consultarse en el Anexo IV: Código de MATLAB.

6.7.6 Movimiento del robot

Una vez calculadas las configuraciones mediante la cinemática inversa y haciendo uso de la API propia, se mandan los comandos necesarios para que el robot ejecute los movimientos. Para mandar los comandos y así conseguir el movimiento del robot es necesario realizar los pasos mostrados en la Figura 75.

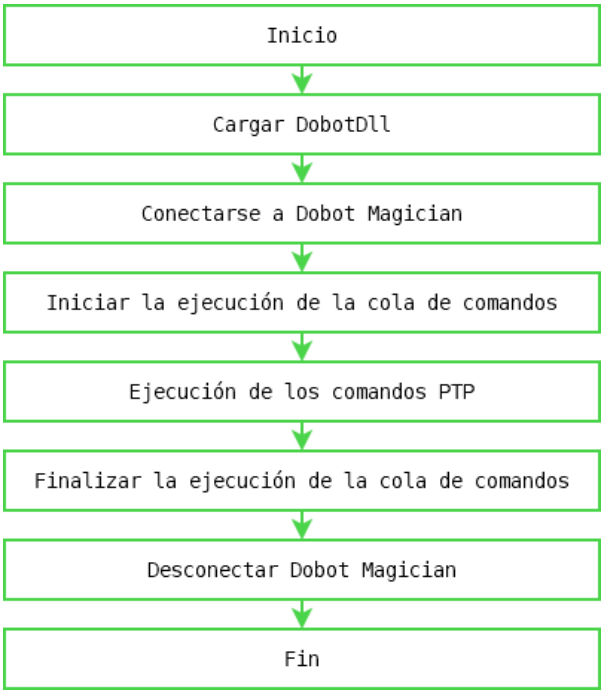


Figura 75. Secuencia para la ejecución de los comandos

No obstante, antes de mandar los comandos es necesario tener en cuenta los sistemas de referencia y unidades en los que trabaja el robot y los que se utilizan en el modelo.

El sistema de referencia tomado para definir la posición de cada unión tiene que ser el mismo en el modelo que en el robot. De no ser así, habrá que aplicar las correspondientes transformaciones para que el valor de la posición sea el correcto.

Para evitar este problema, en el modelado del sistema se ha procurado que los sistemas de referencia coincidan con los utilizados en el robot. A pesar de ello, el fabricante utiliza una forma de referenciar la posición del antebrazo que no es posible modelar directamente. Concretamente para establecer la posición del antebrazo el robot utiliza el ángulo con la horizontal. Por el contrario, en el sistema modelado dicha posición se define como el ángulo del antebrazo respecto del brazo (ver Figura 76).

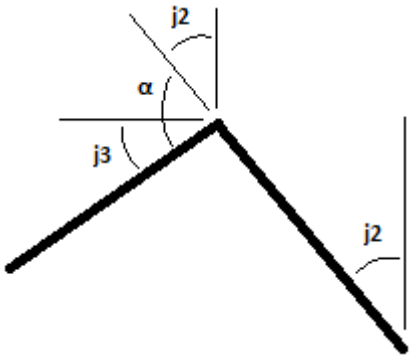


Figura 76. Relación entre el ángulo del antebrazo calculado (α) y el utilizado por el robot (j3).

Por tanto, el ángulo $j3$ utilizado por el robot se calcula según la siguiente ecuación:

$$j3 = \alpha + j2 - 90° \quad \textbf{(13)}$$

Donde α es el ángulo q_3 (ver apartados 6.7.2 y 6.7.3).

En cuanto a las unidades utilizadas, el robot utiliza grados para los ángulos y milímetros para las distancias mientras que en el modelo se utilizan radianes y metros.

A continuación, se explican las funciones elaboradas para facilitar el control del robot:

- **LoadDobotModel**

En esta funcion se define el modelo cinemático del robot mediante la *Robotics System Toolbox*. También se añaden los modelos 3D a cada eslabón para su posterior visualización.

Como parámetro de salida devuelve un árbol de objetos que define el robot. Esta estructura de datos es utilizada por la *Robotics System Toolbox* como base para todos los cálculos. Contiene información sobre los eslabones y uniones. Los nombres que se les ha asignado son los siguientes:

Eslabones:
‘base’: Eslabón fijo o base.
‘shoulder’: Eslabón correspondiente al hombro.

'arm': Eslabón correspondiente al brazo.
'forearm': Eslabón correspondiente al antebrazo.
'wrist': Eslabón correspondiente a la muñeca.
'hand': Eslabón correspondiente a la mano.
'effector': Sirve para posicionar el efector final.

Uniones:
'q1': Articulación del hombro.
'q2': Articulación del brazo.
'q3': Articulación del antebrazo.
'q4': Articulación de la muñeca.
'q5': Articulación de la mano.

Esta función no tiene ningún parámetro de entrada.

En ciertas ocasiones puede ser necesario modificar la posición del efector final según la mano protésica que se acople al brazo y según el tipo de agarre que se quiera realizar. En esta función pueden modificarse fácilmente los valores de las coordenadas del efector final (w, h) con respecto al sistema de referencia S_4 (ver Figura 73).

- **DirectKinem**

Aunque la *Robotics System Toolbox* permite la obtención de la posición del efector final dado el ángulo de cada articulación, se han agrupado todas las líneas de código necesarias en una única función para agilizar la obtención de la cinemática directa.

Como parámetros de entrada tiene un árbol de objetos que define el robot (ver *LoadDobotModel*) y una postura. Esta postura se define como un vector fila de 5 estructuras. Cada estructura cuenta con 2 campos, 'JointName' y 'JointPosition' los cuales contienen el nombre de la unión (ver *LoadDobotModel*) y el ángulo girado por la misma en radianes respectivamente.

Como dato de salida devuelve un vector con las coordenadas del efector final (x, y, z) para los ángulos de cada articulación especificados en la postura.

- **InvKinem**

Permite la obtención de los ángulos de las articulaciones necesarios para llevar el efector al punto especificado.

Como parámetros de entrada tiene un árbol de objetos que define el robot (ver *LoadDobotModel*), un vector con las coordenadas (x, y, z) del efector final en el sistema de referencia global y el ángulo de giro de la mano (q5) en radianes.

Como resultado devuelve una postura del robot. Para ver como se define esta postura consultar *DirectKinem*.

- **AddTrayectoryToQueue**

Añade uno o varios comandos a la cola para que el robot pase por las posturas especificadas.

Como dato de entrada tiene una matriz de tamaño n x 5 siendo n el número de posturas por las que debe pasar. Cada fila será, por tanto, un vector de 5 estructuras correspondiente a una postura. Para ver como se define esta postura consultar *DirectKinem*.

Como dato de salida devuelve un entero que indica la posición del último comando añadido a la cola.

Los comandos de movimiento enviados desde esta función son del tipo MOVJ_ANGLE (ver apartado 2.1.2).

- **MoveDobot**

En esta función se gestionan todos los pasos necesarios para mover el brazo robot.

Como datos de entrada tiene una matriz de tamaño n x 4 siendo n el número de puntos por los que debe pasar el efector final y un parámetro que indica si se quiere simular el movimiento o mandar los comandos al robot para que los ejecute. Si dicho parámetro es 0 (o falso), se mandarán los comandos al robot para realizar los movimientos, si es 1 (o verdadero), se mostrará una ventana con la simulación del movimiento y no se mandará ningún comando al robot.

Cada punto de la trayectoria se define como un vector fila con las coordenadas cartesianas en mm seguidas del ángulo de la mano en radianes (x, y, z, q5).

Tanto si se está simulando el movimiento o mandando los comandos al robot, previamente esta función comprueba si los ángulos de cada articulación están dentro de los límites especificados (ver *LoadDobotModel*). En caso negativo, la función dejará de ejecutarse devolviendo un error como resultado. Este error indica que articulación ha sobrepasado los límites y en qué coordenadas se ha producido el error.

Como salida devuelve una cadena de imágenes con la que se puede volver a visualizar la simulación mediante el comando *movie* de *MATLAB*.

Si se desea cambiar la configuración por defecto para el tiempo que dura la simulación bastará con cambiar el parámetro que se muestra a continuación dentro de la función. Este parámetro también afecta a la resolución de la simulación, a mayor tiempo, mayor número de imágenes en la simulación.

- 'tFinal': Tiempo total de la simulación (10 s por defecto)

- **GetJacobian**

Aunque no se ha implementado el control de la velocidad, se ha elaborado esta funcion que devuelve el jacobiano calculado para unas coordenadas articulares $(q1, q2, q4, q5)$ dadas. Para más información sobre el cálculo del jacobiano consultar el apartado 6.7.4.

- **GetEndEffectorSpeeds**

Haciendo uso de la función *GetJacobian*, esta función devuelve la velocidad del efector final en coordenadas cartesianas dados la velocidad y posición $(q1, q2, q4, q5)$ de las articulaciones.

- **GetJointSpeeds**

También hace uso de la función *GetJacobian*. Devuelve la velocidad de las articulaciones dados la velocidad del efector final en coordenadas cartesianas y la posición $(q1, q2, q4, q5)$ de las articulaciones.

El código de todas las funciones descritas anteriormente se puede consultar en el Anexo IV: Código de MATLAB.

6.7.7 Ejemplo de control

Con todas las funciones desarrolladas se ha elaborado un ejemplo de una secuencia de comandos que permite simular o mover el robot.

Primero se carga el árbol de elementos para poder obtener la configuración de reposo definida. Esta posición se corresponde con un valor $(0°, 45°, -45°, 0°)$ de las coordenadas articulares (q_1, q_2, q_3, q_5).

```
dobot = LoadDobotModel;

home_pos = DirectKinem(dobot,dobot.homeConfiguration);
```

Después, se define los puntos por los que debe pasar el efector final del robot. Cada punto se define como un vector fila con las coordenadas cartesianas seguidas del ángulo de la mano (x, y, z, q_5). En este caso, el primer punto al que se desplazará el robot será a la posición de reposo.

```
waypoints = [home_pos 0;...
             420 -20 20 0;...
             400 -50 100 pi/4;...
             390 -150 100 pi/2;...
             370 -180 50 pi/2;...
             370 -190 -70 pi/2;...
             370 -170 -70 pi/2];
```

A continuación, se llama a la función *MoveDobot* pasando el segundo parámetro como verdadero, indicando que se quiere simular el movimiento. Finalmente, mediante el comando *movie* de *MATLAB* se reproduce la animación de la simulación. En caso de pasar el segundo parámetro como falso, se enviarán los comandos para que el robot realice el movimiento. En esta situación sería necesario eliminar la línea que muestra la simulación puesto que dará error al no haber ninguna simulación que mostrar.

```
frames = MoveDobot(waypoints,true);

movie(frames,1,15) %Visualizar la animación resultante de la simulación
```

En la Figura 77 se muestra el resultado de la simulación, donde puede verse en azul la trayectoria seguida por el efector final.

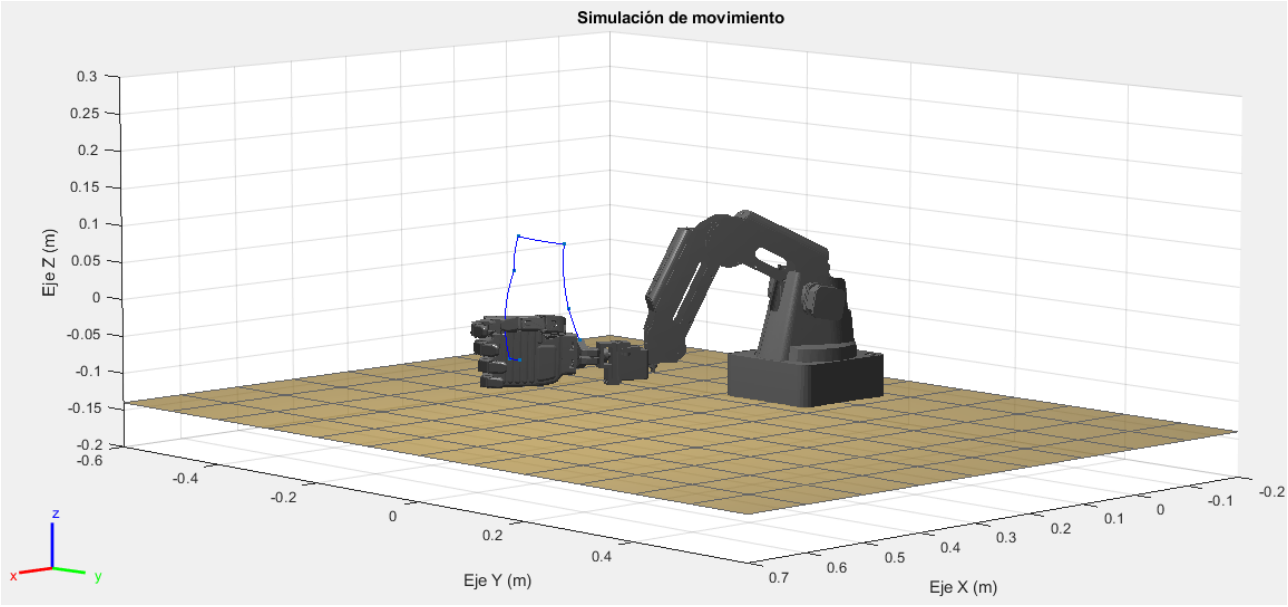


Figura 77. Ejemplo 1 del control

En caso de querer realizar otro tipo de movimientos o si se necesita un mayor control que el proporcionado por la función *MoveDobot* consultar la información referente al resto de funciones en el Anexo IV: Código de MATLAB. Además, en dicho anexo, se pueden encontrar otros ejemplos de trayectorias.

6.8 Rango de operación del robot

Por una parte, para evitar el envío de comandos al robot que provoquen configuraciones inválidas del mismo, en la programación de *MATLAB* se ha añadido una comprobación para asegurar que los puntos especificados se encuentran dentro del rango de trabajo del robot. Esta comprobación se basa en comprobar si cada articulación se encuentra dentro de los límites asignados.

Por otro lado, se han obtenido las ecuaciones que definen el volumen de trabajo. El volumen de trabajo se define como los todos los puntos del espacio en los que puede posicionarse le efector final. El efector final puede cambiar su posición dependiendo de la mano protésica instalada, por lo que los cálculos se han realizado en función de sus coordenadas (w, h). Además, el ángulo de la mano también cambia ligeramente el volumen de trabajo (ver Figura 78).

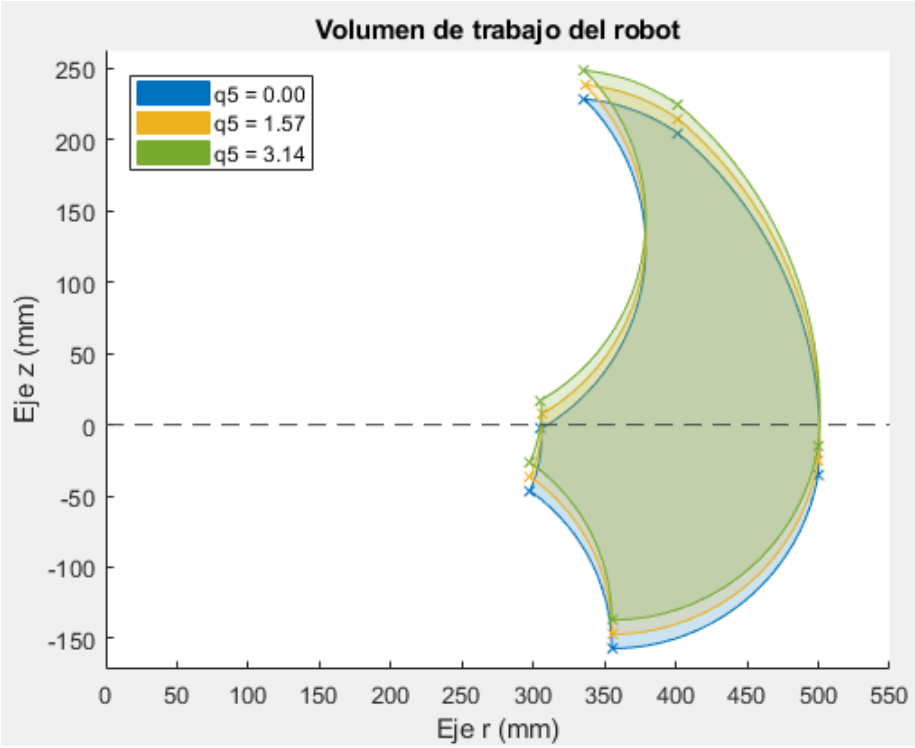


Figura 78. Sección del volumen de trabajo del robot (w=220 mm, h=10 mm).

El volumen de trabajo se ha definido en coordenadas cilíndricas (r, θ, z) puesto que es más fácil de representar dada su simetría. En la Figura 78, se muestra la sección del volumen de trabajo, barriendo esta área desde $\theta = -\frac{\pi}{2}$ hasta $\theta = \frac{\pi}{2}$ se obtiene el volumen de trabajo completo. Las ecuaciones que definen esta sección del volumen de trabajo pueden consultarse en el Anexo II: Cálculo del volumen de trabajo.

7 VIABILIDAD ECONÓMICA

Al tratarse de un proyecto que forma parte de una linea de investigación, no se trata de un producto comercial y por tanto no existe una competencia en el mercado.

Se podría estudiar la viabilidad económica del proyecto comparando con investigaciones similares. No obstante, se han encontrado varios problemas, el primero es que no se han encontrado investigaciones comparables. El segundo es que, en los artículos de investigación, no se suele proporcionar presupuesto o precio de los componentes utilizados.

Por ello, se ha optado por hacer un estudio económico comparando los costes de utilizar un software u otro para realizar el proyecto.

Tabla 7.1. Comparación de los costes según software utilizado

Descripción	*Lenguaje alto nivel*	*MATLAB + Robotics System Toolbox*	*LabVIEW*
Precio del software	0	1000	3000
Coste de las horas trabajo de desarrollo de la programación	500 h -> 5500 €	300 h -> 3300 €	300h -> 3300 €
Total	5500	4300	6300

Para consultar el coste total estimado del proyecto y el desglose de los costes ver el documento PRESUPUESTO.

8 CONCLUSIONES

Con el trabajo presente se ha conseguido diseñar y fabricar las piezas necesarias para acoplar manos protésicas como las desarrolladas por el grupo de Biomecánica y Ergonomía de la Universitat Jaume I al brazo robot comercial *Dobot Magician*. Además, se ha programado un control de posición del brazo robot en *MATLAB*, el cual permite la simulación virtual previa del movimiento.

Mediante la incorporación de las manos protésicas a un brazo robot se permite la realización de ensayos y estudios más parecidos a la realidad, facilitando la caracterización del desempeño de las manos.

El control desarrollado tiene como dato de entrada los puntos de paso del efector final para definir el movimiento del robot, es decir, no se puede modificar la velocidad ni aceleración durante la trayectoria. No obstante, se ha elaborado la estructura y funciones base para poder desarrollar un control más avanzado en el futuro de forma relativamente sencilla.

9 BIBLIOGRAFÍA

[1] DevalHand, https://bit.ly/2GIpleU, 12 de febrero de 2019

[2] BENCH-HAND, http://bit.ly/34byd4H, 11 de diciembre de 2019

[3] I. Llop-Harillo & A. Pérez-González (2017) System for the experimental evaluation of anthropomorphic hands. Application to a new 3D-printed prosthetic hand prototype, International Biomechanics, 4:2, 50-59.

[4] F.J. Andrés, A. Pérez-González, C. Rubert, J. Fuentes, B. Sospedra. Comparison of grasping performance of tendon and linkage transmission systems systems in an electric powered low cost hand prosthesis. Journal of Mechanisms and Robotics, 11(1): 011018, doi:10.1115/1.4040491

[5] Dobot Magician, http://bit.ly/2PBE7He, 12 de febrero de 2019

[6] Dobot – Dobot Magician – Download center – Product manual, https://bit.ly/2BN6491, 12 de febrero de 2019

[7] RobinHsieh – InMoov Arm, https://bit.ly/2BB1omc, 15 de febrero de 2019

[8] Dobot – Dobot Magician – Specifications, https://bit.ly/2GHloXI, 12 de febrero de 2019

[9] RobotShop, https://bit.ly/2NbRBb6, 15 de febrero de 2019

[10] Pololu, https://bit.ly/2S4t0FJ, 15 de febrero de 2019

[11] Denavit-Hartenberg, http://bit.ly/35fThIF, 20 de octubre de 2019

[12] Matriz pseudoinversa de Moore-Penrose, http://bit.ly/2rD1vMu, 15 de noviembre de 2019

ANEXO I: ANÁLISIS DE RESISTENCIA DE LAS PIEZAS

Para los análisis estáticos se ha considerado unas coordenadas (w, h) del efector final $(220, 50)$ cómo puede verse en la Figura 79. La carga que se considerará será la carga máxima del robot aplicada en el punto correspondiente al efector final, la cual tiene un valor de 0,5 Kg. Se ha elegido este punto considerando que la mayor parte del peso estará situada en el centro de gravedad del objeto sujetado. En la realidad, la mano representará una parte importante de dicho peso por lo que el centro de gravedad del conjunto (mano y objeto sujetado) siempre estará más cercano a la muñeca que en el caso estudiado y por tanto los momentos que aparecerán serán menores. De esta forma, se asegura que el caso calculado es más desfavorable que el real.

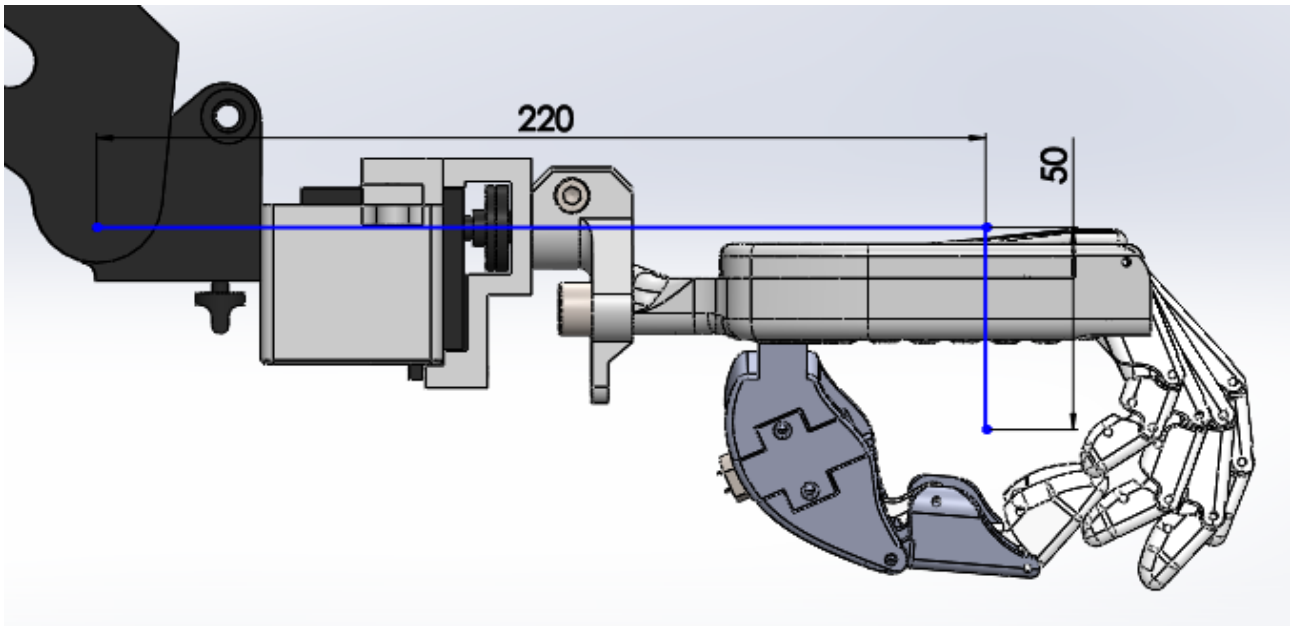


Figura 79. Coordenadas del efector final (w, h)

En cuanto al material se ha creado un nuevo material en *SolidWorks* con las características del PLA. Las características utilizadas en los análisis son las mostradas a continuación:

Módulo de Young	3,5	GPa
Coeficiente de Poisson	0,39	
Densidad	1240	kg/m^3
Límite elástico	50	MPa

- **Acoplador de la mano**

Primero se ha calculado la fuerza necesaria de apriete de la pieza sobre el eje del servo para evitar que el acoplador se deslice. Partiendo de la ecuación (14) se despeja la fuerza de apriete N. (ver Figura 80)

$$F_r = \mu \cdot N \rightarrow N = \frac{F_r}{\mu} \quad \textbf{(14)}$$

*Donde F_r es la fuerza de rozamiento máxima a partir de la cual se producirá deslizamiento, μ es el coeficiente de rozamiento y N es la fuerza normal.

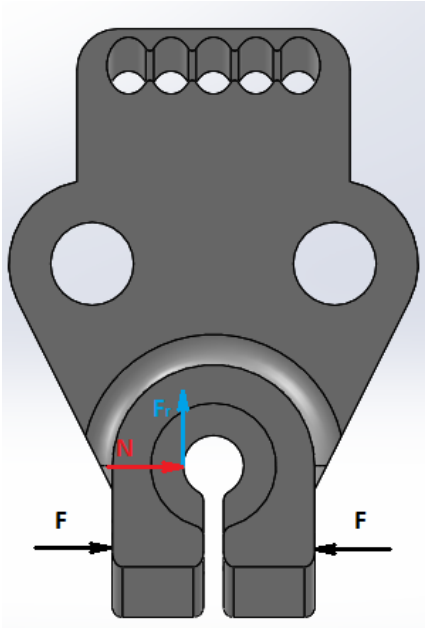


Figura 80. Carga aplicada sobre el acoplador de la mano

El coeficiente de rozamiento del PLA se ha estimado a partir del valor del coeficiente de rozamiento del ABS puesto que no se ha encontrado un valor en la bibliografía. No obstante, se ha encontrado que el coeficiente de rozamiento en el PLA es superior que en el ABS. Por tanto, tomando el valor del ABS ($\mu = 0.5$) se estará del lado de la seguridad.

Para obtener F_r se calcula el momento M que la carga F aplicará sobre el eje.

$$M = F \cdot d = 4{,}9\ N \cdot 50\ mm = 245\ N \cdot mm \quad \textbf{(15)}$$

*Siendo d la distancia a la que se aplica la carga.

Conocido el momento, puede calcularse la fuerza tangencial F_t a la que el eje estará sometido.

$$F_t = \frac{M}{r} = \frac{245\ N \cdot mm}{3\ mm} = 81{,}7\ N \quad \textbf{(16)}$$

*Siendo r el radio del eje.

La fuerza de rozamiento mínima necesaria para evitar el deslizamiento será, por tanto, la fuerza tangencial obtenida.

$$F_r = F_t \quad \textbf{(17)}$$

Finalmente, pueden sustituirse los valores en la ecuación (14) para obtener la fuerza normal necesaria.

$$N = \frac{81{,}7\ N}{0{,}5} = 163{,}4\ N$$

No obstante, la fuerza real aplicada será superior debido a que la carga puede ser ligeramente superior debido a efectos dinámicos y, además, al realizar el montaje se realizará un apriete mayor al necesario para asegurar que el acoplador no deslice. Por estos motivos, a la fuerza normal obtenida se le ha aplicado un coeficiente de mayoración de 2.

$$N = 163{,}4\ N \cdot 2 = 327\ N$$

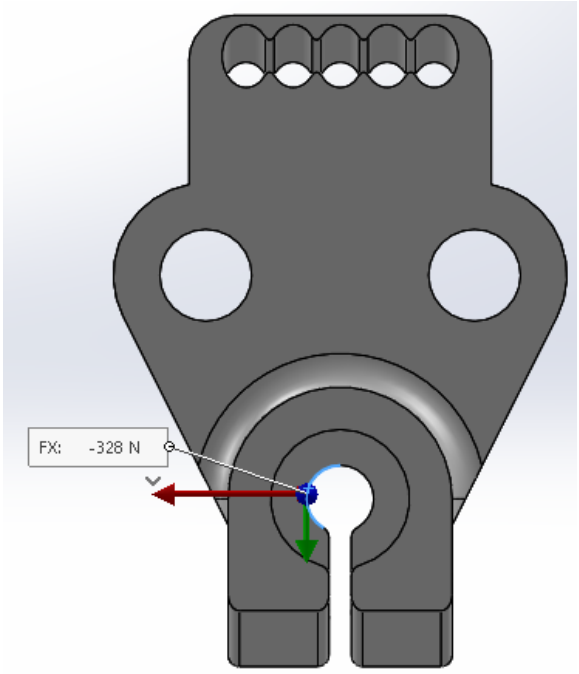


Figura 81. Fuerza de reacción debida al apriete del tornillo

La fuerza normal obtenida corresponde a la parte de la pieza en contacto con el eje.

Una vez obtenida la fuerza normal, se puede calcular la fuerza necesaria de apriete del tornillo que provoca dicha fuerza normal. Para calcular el par de apriete del tornillo M_a, se ha configurado una conexión por tornillo en *SolidWorks* y se ha ido tanteando el valor de par de apriete hasta conseguir que la fuerza de reacción en el eje sea similar a la calculada (ver Figura 81).

$$M_a = 0{,}76\ Nm$$

Para la aplicación de la carga de 0,5 Kg se han considerado 4 casos según la dirección en la que se aplica.

En el primero se considera que la carga se aplica verticalmente hacia abajo, simulando el agarre de un objeto con la palma de la mano orientada hacia abajo.

El segundo caso considera una fuerza lateral, simulando un agarre con la mano posicionada lateralmente (Por ejemplo, el agarre de una botella).

El tercer y cuarto caso consideran una carga axial de compresión y tracción respectivamente. Estos simulan el empuje o arrastre de un objeto.

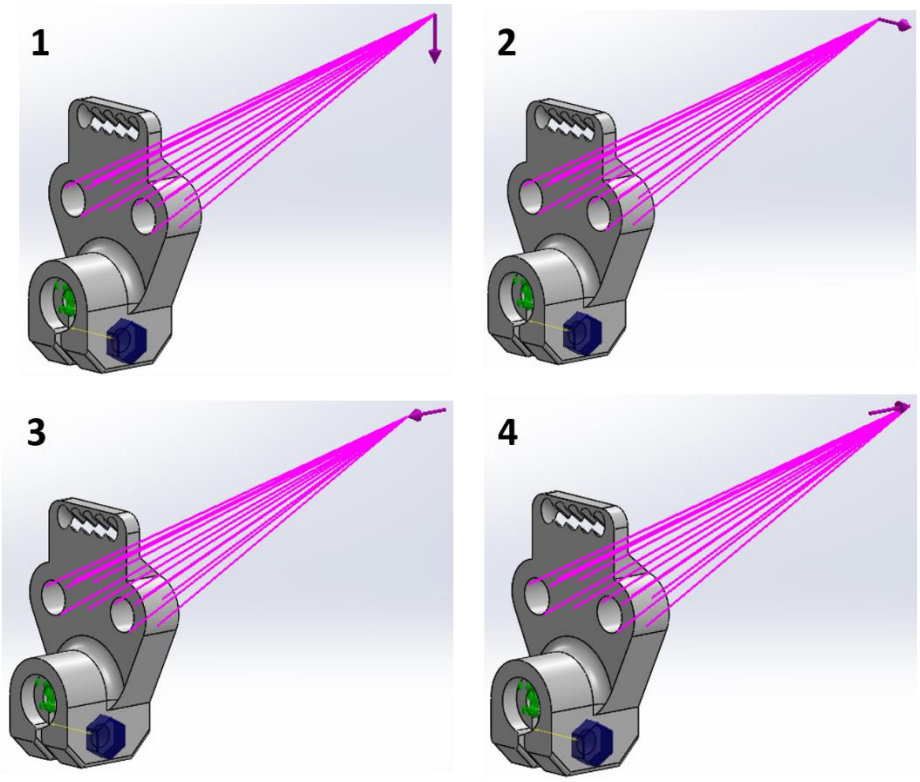


Figura 82. Casos de cargas en el análisis del acoplador de la mano

El mallado realizado para todas las piezas ha sido el mismo. Se ha considerado un tamaño de elemento de 1,5 mm. (ver Figura 83).

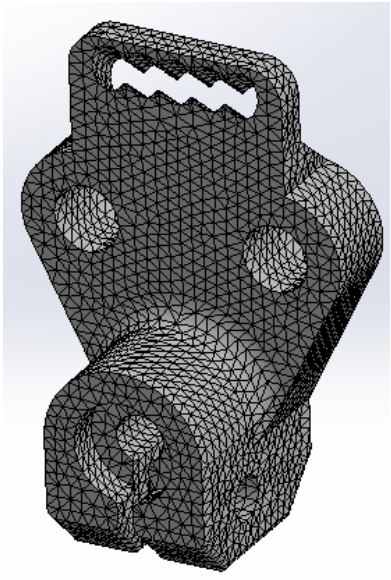
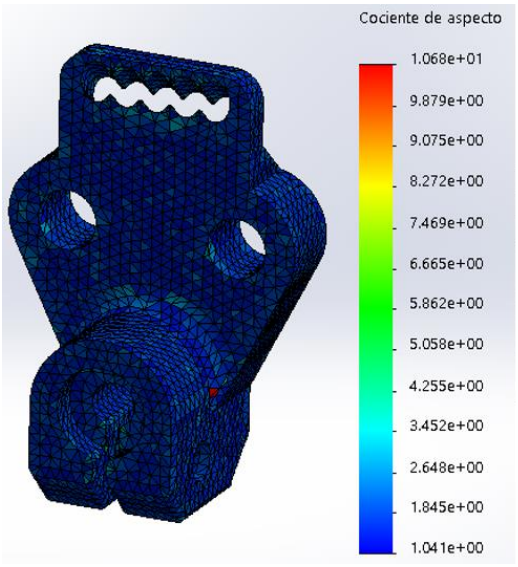


Figura 83. Mallado del acoplador de la mano

Para comprobar la validez de este mallado se ha realizado un trazado del coeficiente de aspecto de los elementos como se puede ver en la Figura 84. Se observa que el 99,3% de los elementos tienen un coeficiente de aspecto menor a 3, lo que quiere decir que prácticamente la totalidad de los elementos están poco deformados y por lo tanto los resultados obtenidos serán fiables.



Número total de nodos	59104
Número total de elementos	37427
Cociente de aspecto máximo	10.682
Porcentaje de elementos con cociente de aspecto < 3	99.3
Porcentaje de elementos con cociente de aspecto > 10	0.00534
% de elementos distorsionados (Jacobiano)	0

Figura 84. Trazado de la calidad de la malla del acoplador de la mano

El resultado de tensiones para los cuatro casos de carga es muy similar. Esto se debe a que la fuerza que más tensiones introduce en la pieza es la correspondiente al apriete del tornillo. Las tensiones máximas en todos los casos se encuentran en el alojamiento del eje del servo según se muestra en la Figura 85.

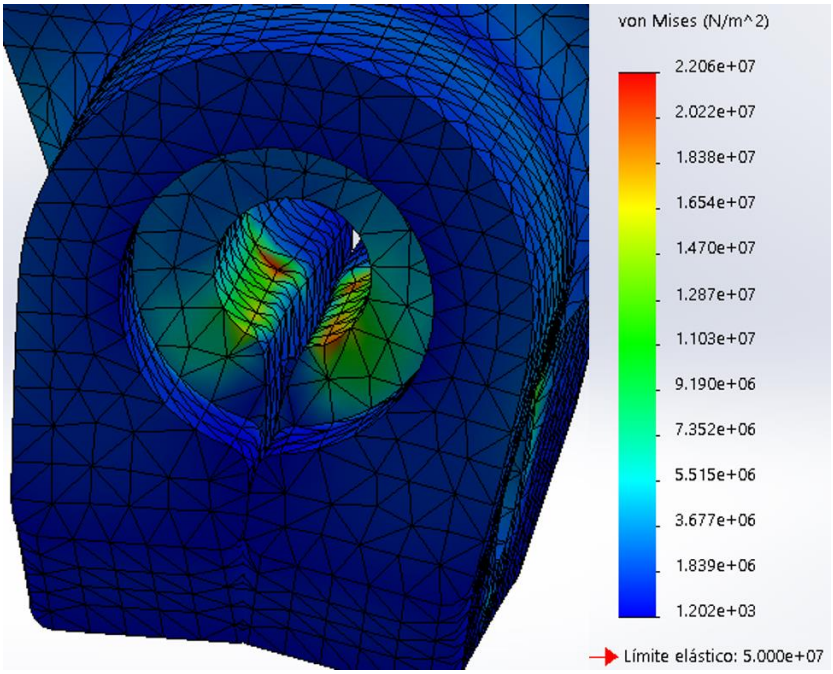


Figura 85. Tensiones máximas en el acoplador de la mano

Para optimizar el material utilizado en la pieza se ha ido cambiando el espesor de pared de la pieza (ver Figura 86) hasta obtener un factor de seguridad con un valor entre 1,5 y 2,5.

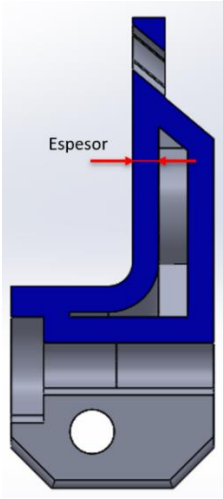


Figura 86. Espesor de pared del acoplador de la mano

El factor de seguridad que finalmente se ha obtenido para todos los casos de carga ha sido de 2,2 correspondiente a un espesor de pared de 3,2 mm.

En cuanto al desplazamiento obtenido para todos los casos es de magnitud similar. (ver Figura 87)

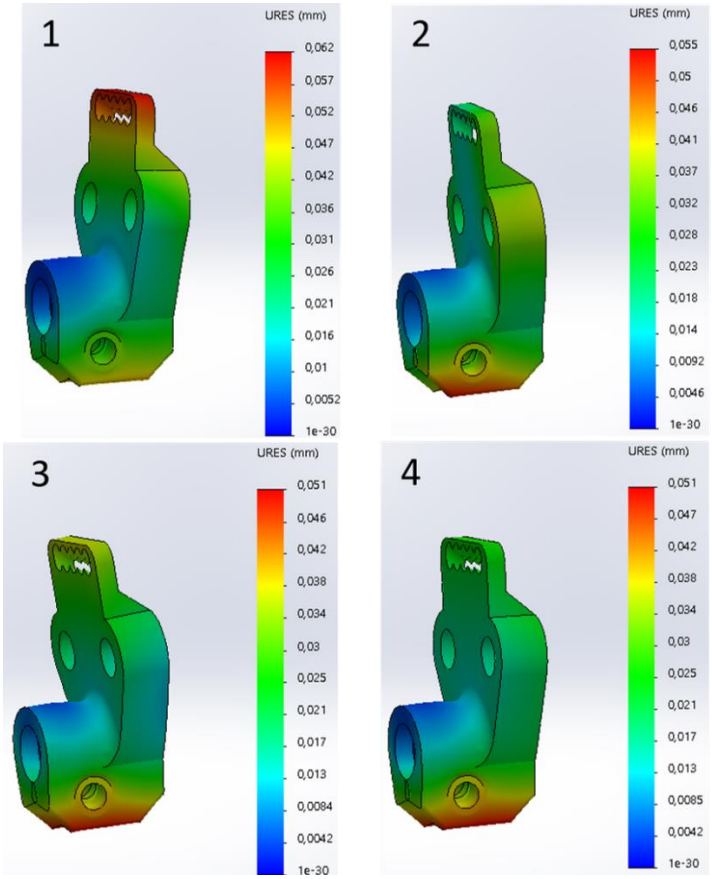


Figura 87. Deformación en el acoplador de la mano

- **Puente del servo y soporte del servo**

Para calcular las tensiones y deformaciones a las que estarán sometidas las piezas del puente del servo y del soporte del servo se realizará un análisis del ensamblaje de estas dos piezas y el servo de forma conjunta (ver Figura 88).

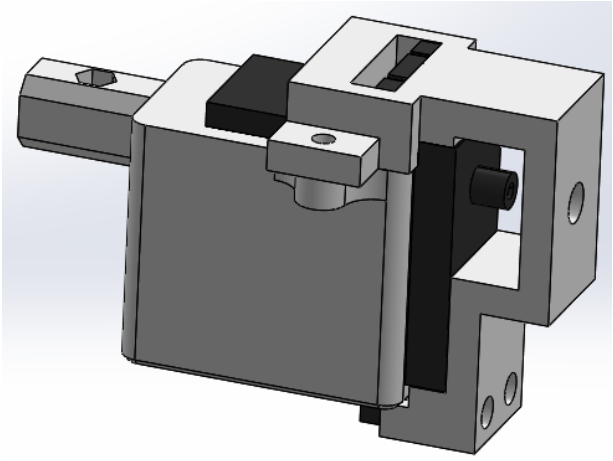


Figura 88. Ensamblaje para el análisis del puente del servo y del soporte del servo

Antes de realizar el análisis se han calculado las reacciones que aparecerán en el servo y en el puente del servo debidas a la carga.

En este análisis se sustituye la carga remota correspondiente al peso de la mano y del objeto que sostenga por un par de fuerzas equivalentes. La fuerza se aplicará, por un lado, en el puente del servo (F_1) y, por otro, en el eje del servo (F_2).

Aplicando equilibrio de momentos respecto al punto de aplicación de F_1 se calcula F_2

$$F \cdot d_1 = F_2 \cdot d_2 \rightarrow F_2 = F \cdot \frac{d_1}{d_2} = 4{,}9\ N \cdot \frac{112\ mm}{10{,}2\ mm} = 54\ N \quad \textbf{(18)}$$

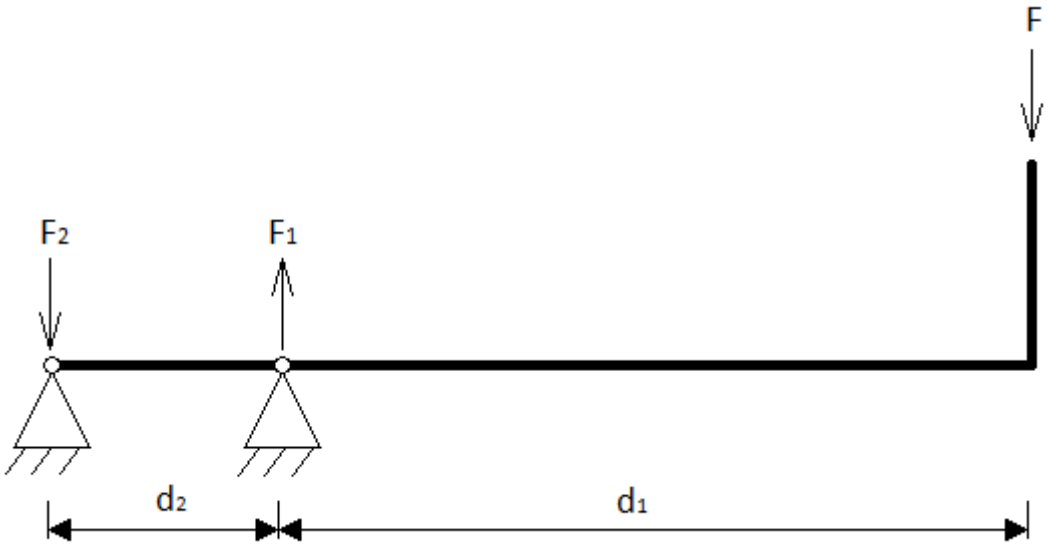


Figura 89. Cálculo de fuerzas para el análisis estático del puente del servo y del soporte del servo

Una vez obtenido F_2, puede calcularse F_1 aplicando equilibrio de fuerzas.

$$F_1 = F_2 + F = 54\ N + 4{,}9\ N = 59\ N \quad \textbf{(19)}$$

Se contemplan 2 casos de cargas, en el primero, la carga se corresponde con un agarre de un objeto con la palma de la mano hacia abajo. En el segundo caso, la carga se corresponde a un agarre con la palma orientada lateralmente (Por ejemplo, agarre de una botella).

En el primer caso solo se considera como cargas el par de fuerzas calculado previamente mientras que en el segundo también se introduce un momento en el eje del servo.

En ambos casos el servo se ha considerado como un elemento rígido y se ha utilizado el mismo mallado para las piezas. El tamaño de elemento elegido es de 2 mm (ver Figura 90).

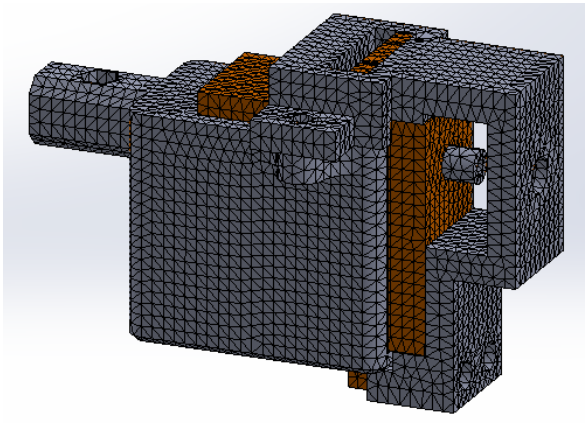


Figura 90. Mallado del puente del servo y del soporte del servo

Para comprobar la validez de este mallado se ha realizado un trazado del coeficiente de aspecto de los elementos. Como puede verse en la Figura 91, prácticamente la totalidad de los elementos tienen un coeficiente de aspecto menor a 3, lo que indica que están poco deformados y, por tanto, proporcionarán resultados fiables. Posteriormente se comprueba que las tensiones máximas no estén situadas en alguno de los pocos elementos que tienen una deformación mayor.

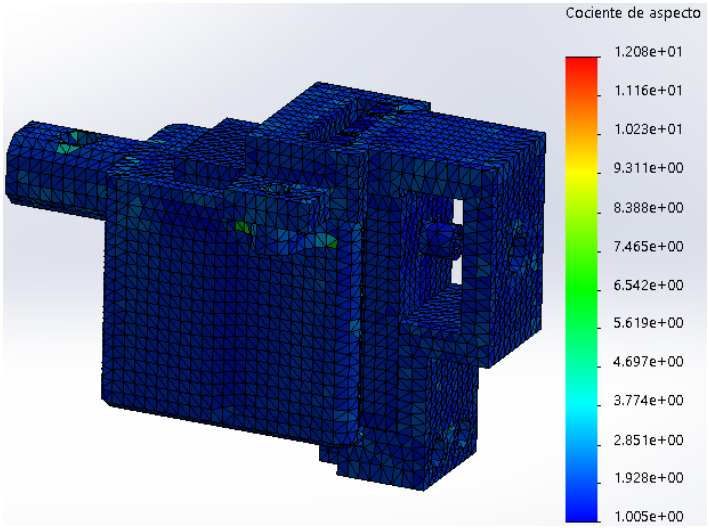


Figura 91. Coeficiente de aspecto de las mallas del puente del servo y del soporte del servo

Tras realizar el análisis de ambos casos, se comprueba que el segundo caso en el que se incluye el momento es el más desfavorable, presentando mayores tensiones como puede verse en la Figura 92).

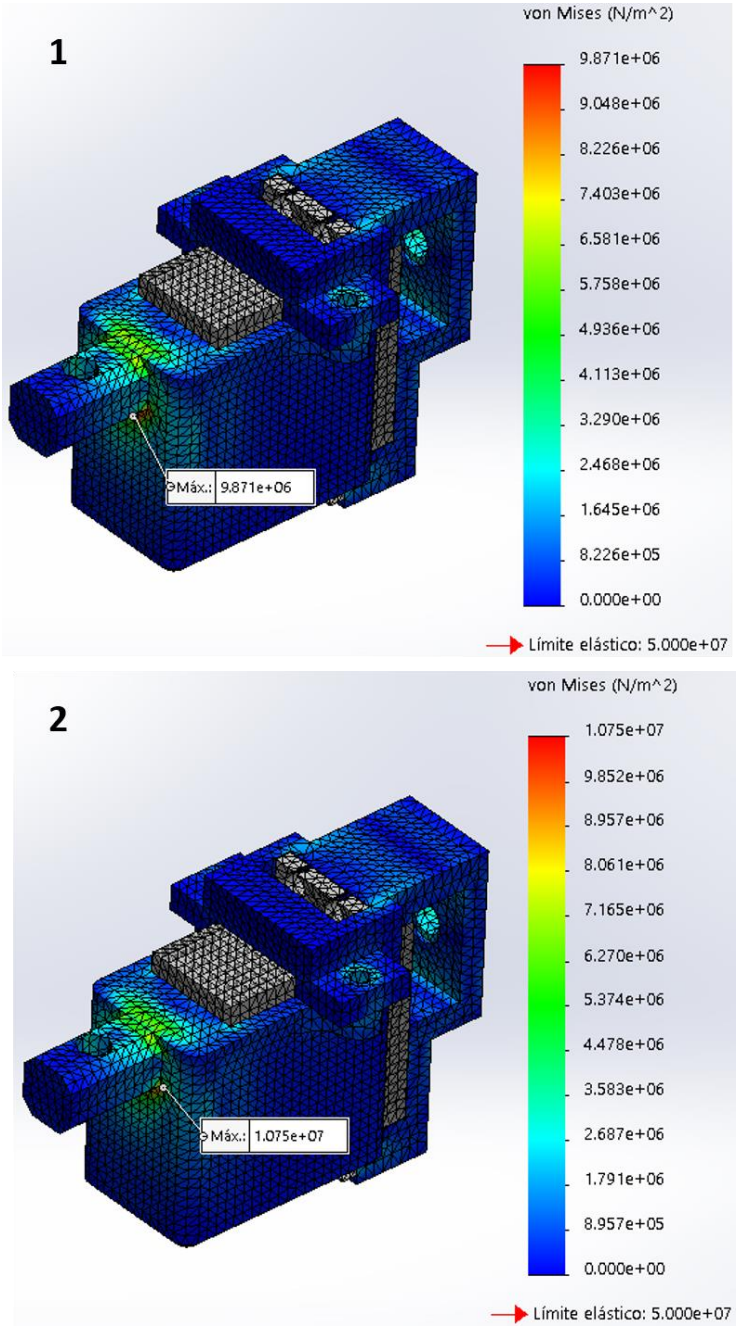


Figura 92. Tensiones en el soporte del servo y el puente del servo

Para el puente del servo no se ha tomado el factor de seguridad como criterio para el diseño ya que se quiere limitar su deformación para que el efector final se desplace lo menos posible al aplicar la carga. Por este motivo la pieza no será hueca como las demás, sino que será maciza.

Para determinar el espesor de pared necesario en el soporte del servo (ver Figura 93) se ha establecido un límite máximo de 1,5 mm para el desplazamiento del efector final al aplicar la carga.

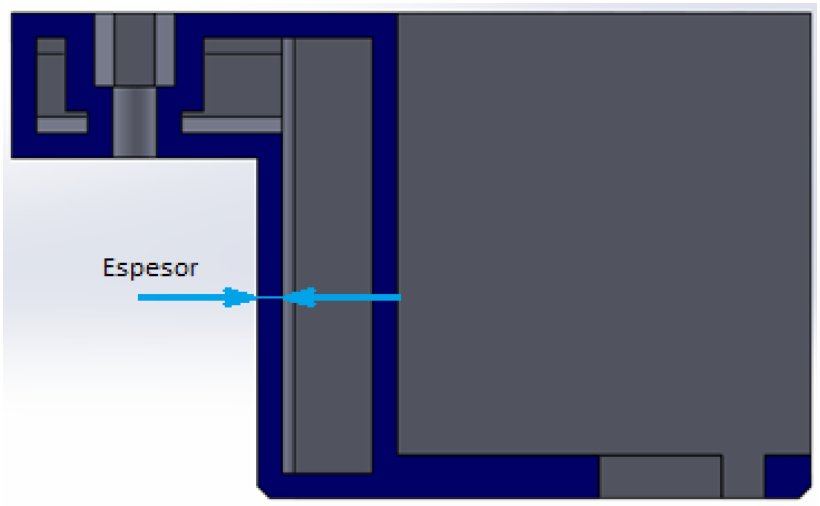


Figura 93. Espesor de pared del soporte del servo

Considerando que la aportación de la deformación del puente del servo es mucho menor y por lo tanto despreciable respecto a la del soporte del servo, se ha calculado el desplazamiento máximo permisible del agujero del puente del servo por semejanza de triángulos tal y como se muestra en la Figura 94.

$$\frac{1{,}5\ mm}{180\ mm} \cdot 65\ mm = 0{,}54\ mm$$

Figura 94. Desplazamiento máximo del agujero del puente del servo

A partir de este valor se ido iterando el diseño del soporte del servo variando su espesor de pared hasta obtener un valor de desplazamiento del agujero del puente del servo inferior a 0,54 mm. (ver Figura 95).

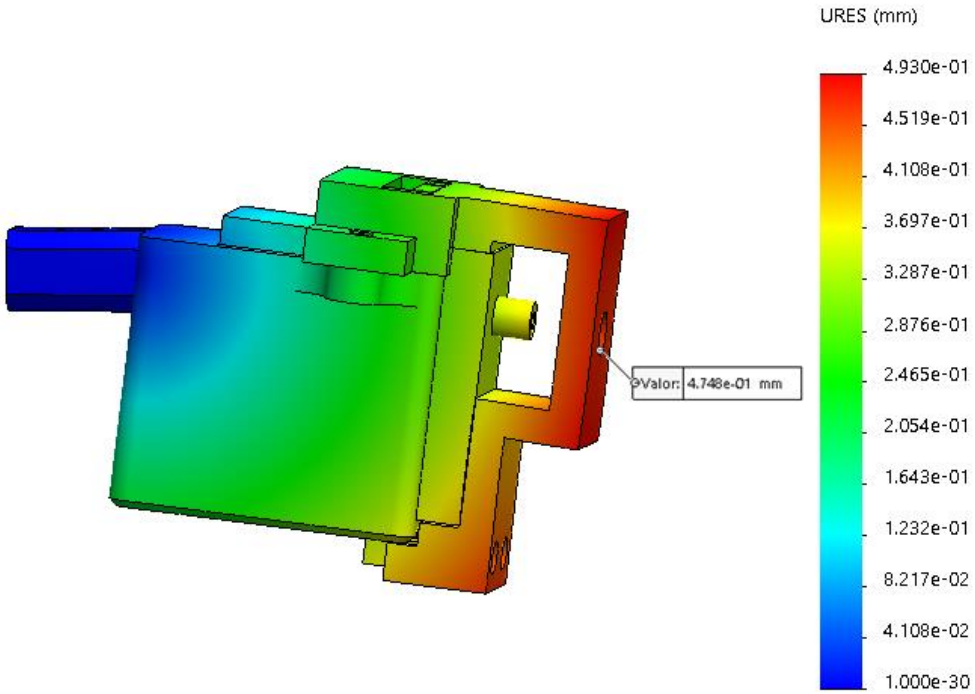


Figura 95. Desplazamiento del agujero del soporte del servo

Finalmente, el espesor de pared obtenido para el soporte del servo es de 2,8 mm.

ANEXO II: CÁLCULO DEL VOLUMEN DE TRABAJO

Las ecuaciones que definen cada tramo del contorno (ver Figura 96) de la sección del volumen de trabajo son las siguientes:

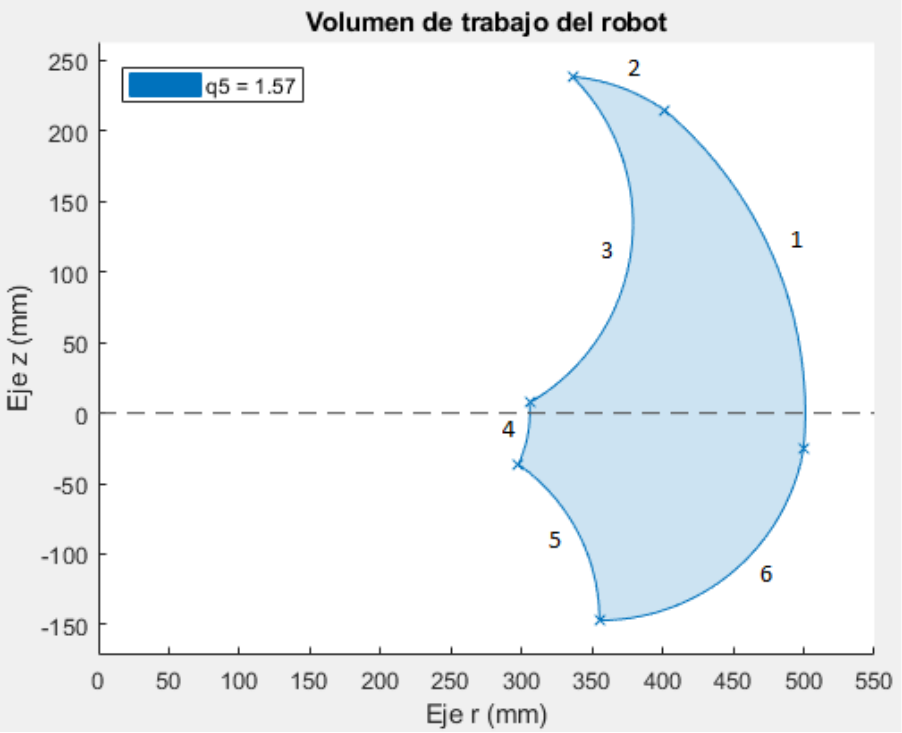


Figura 96. Tramos del contorno de la sección del volumen de trabajo

Para obtener las ecuaciones se ha supuesto el ángulo $q_1 = 0$ puesto que la sección obtenida para todos los valores de $q_1$será la misma. De esta forma el valor de la coordenada y es:

$$y = h \cdot \operatorname{sen} q_5 \tag{20}$$

Para pasar las ecuaciones posteriores a coordenadas cilíndricas se sustituye la variable x según la ecuación (21).

$$r = \sqrt{x^2 + y^2} \rightarrow x = \sqrt{h \cdot \operatorname{sen} {q_5}^2 - r^2} \tag{21}$$

Tramo 1:

$$z = \pm\sqrt{(l_2 + l_3 \cdot \cos 10°)^2 + (l_3 \cdot \operatorname{sen} 10°)^2 - (x - w)^2} - h \cdot \cos q_5 \tag{22}$$

Tramo 2:

$$z = l_2 \cdot \sqrt{1 - \left(\frac{x - (l_3 \cdot \cos 45° + w)}{l_2}\right)^2} + l_3 \cdot \operatorname{sen} 45° - h \cdot \cos q_5 \tag{23}$$

Tramo 3:

$$z = \pm\sqrt{(l_3 \cdot \cos 45°)^2 + (l_3 \cdot \operatorname{sen} 45°)^2 - (x - w - l_2 \cdot \cos 85°)^2} + l_2 \cdot \operatorname{sen} 85° - h \cdot \cos q_5 \quad \textbf{(24)}$$

Tramo 4:

$$z = -\sqrt{(l_2 \cdot \cos 85° + l_3 \cdot \cos 60°)^2 + (l_2 \cdot \operatorname{sen} 85° - l_3 \cdot \operatorname{sen} 60°)^2 - (x - w)^2} - h \cdot \cos q_5 \quad \textbf{(25)}$$

Tramo 5:

$$z = l_2 \cdot \sqrt{1 - \left(\frac{x - (l_3 \cdot \cos 90° + w)}{l_2}\right)^2} - l_3 \cdot \operatorname{sen} 90° - h \cdot \cos q_5 \quad \textbf{(26)}$$

Tramo 6:

$$z = -\sqrt{l_3^2 - (x - w - l_2)^2} - h \cdot \cos q_5 \quad \textbf{(27)}$$

ANEXO III: CÁLCULO DE LA CINEMÁTICA

• Cálculo de la cinemática directa

Para obtener el modelo cinemático directo del brazo robot se ha utilizado el algoritmo de Denavit-Hartenberg.

Primero, se han definido los sistemas de referencia de cada eslabón y se han identificado los distintos elementos siguiendo el método, según se muestra en la Figura 97.

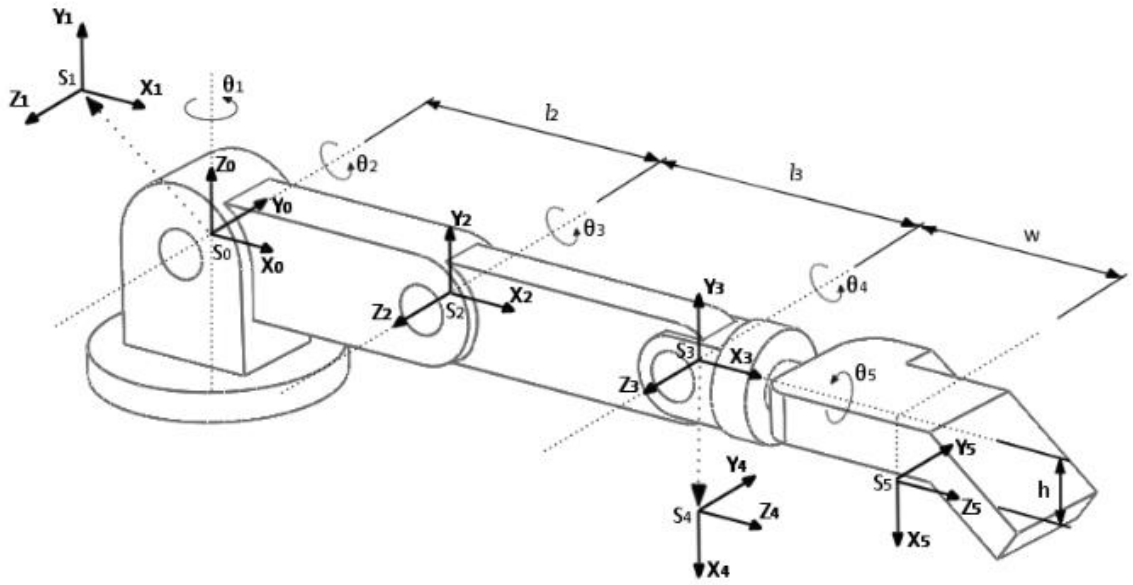


Figura 97. Definición de sistemas de referencia e identificación según Denavit-Hartenberg

Posteriormente se obtienen los 4 parámetros de Denavit-Hartenberg de cada eslabón $(\theta_i, d_i, a_i, \alpha_i)$ que se definen de la siguiente forma:

- θ_i es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_{i-1}, utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.
- d_i es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas $(i-1)$-ésimo hasta la intersección del eje z_{i-1} con el eje x_i. Se trata de un parámetro variable en articulaciones prismáticas.
- a_i es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes z_{i-1} y z_i.
- α_i es el ángulo de separación del eje z_{i-1} y el eje z_i, medido en un plano perpendicular al eje x_i, utilizando la regla de la mano derecha.

Una vez obtenidos pueden calcularse las matrices de transformación homogénea ${}^{i-1}A_i$ mediante la ecuación (28), las cuales definen la posición y orientación de cada eslabón con su consecutivo.

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \cdot sen\,\theta_i & sen\,\alpha_i \cdot sen\,\theta_i & a_i \cdot \cos\theta_i \\ sen\,\theta_i & \cos\alpha_i \cdot \cos\theta_i & -sen\,\alpha_i \cdot \cos\theta_i & a_i \cdot sen\,\theta_i \\ 0 & sen\,\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(28)} $$

Sustituyendo en (28) se obtienen las matrices ${}^0A_1$, ${}^1A_2$, ${}^2A_3$, ${}^3A_4$ y ${}^4A_5$.

$$ {}^0A_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^1A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & l_2 \cdot C_2 \\ S_2 & C_2 & 0 & l_2 \cdot S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & l_3 \cdot C_3 \\ S_3 & C_3 & 0 & l_3 \cdot S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^3A_4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^4A_5 = \begin{bmatrix} C_5 & -S_5 & 0 & h\,C_3 \\ S_5 & C_5 & 0 & h\,S_3 \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Siendo $l_2 = 135\ mm,\ l_3 = 147\ mm,\ h = 10\ mm,\ w = 220\ mm,\ S_i = sen\,\theta_i\ y\ C_i = \cos\theta_i$.

Finalmente, se obtiene la matriz de transformación T, que relaciona la posición y orientación del efector final del robot con respecto a la base en funcion de las coordenadas articulares ($\theta_1,\ \theta_2,\ \theta_3,\ \theta_5$) según la ecuación (29). El ángulo θ_4 está determinado por el mecanismo interno del brazo y depende de los ángulos θ_2 y θ_3 según la ecuación (30).

$$ T = {}^0A_1\,{}^1A_2\,{}^2A_3\,{}^3A_4\,{}^4A_5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(29)} $$

$$ \theta_4 = -(\theta_2 + \theta_3) \quad \textbf{(30)} $$

La terna ortonormal de vectores n, o, a representa la orientación y el vector p representa la posición.

A continuación, se desarrollan los términos de la matriz T.

$$ n_x = S_1\,S_5 + C_5\left(C_4\,(C_1\,C_2\,C_3 - C_1\,S_2\,S_3) - S_4\,(C_1\,C_2\,S_3 + C_1\,C_3\,S_2)\right) $$

$$ n_y = C_5\left(C_4\,(C_2\,C_3\,S_1 - S_1\,S_2\,S_3) - S_4\,(C_2\,S_1\,S_3 + C_3\,S_1\,S_2)\right) - C_1\,S_5 $$

$$ n_z = C_5\left(C_4\,(C_2\,S_3 + C_3\,S_2) + S_4\,(C_2\,C_3 - S_2\,S_3)\right) $$

$$ o_x = C_5\,S_1 - S_5\left(C_4\,(C_1\,C_2\,C_3 - C_1\,S_2\,S_3) - S_4\,(C_1\,C_2\,S_3 + C_1\,C_3\,S_2)\right) $$

$$ o_y = -C_1\,C_5 - S_5\left(C_4\,(C_2\,C_3\,S_1 - S_1\,S_2\,S_3) - S_4\,(C_2\,S_1\,S_3 + C_3\,S_1\,S_2)\right) $$

$$o_z = -S_5\left(C_4\,(C_2\,S_3 + C_3\,S_2) + S_4\,(C_2\,C_3 - S_2\,S_3)\right)$$

$$a_x = S_4\,(C_1\,C_2\,C_3 - C_1\,S_2\,S_3) - C_4\,(C_1\,C_2\,S_3 + C_1\,C_3\,S_2)$$

$$a_y = \mathrm{S_4\,(C_2\,C_3\,S_1 - S_1\,S_2\,S_3) - C_4\,(C_2\,S_1\,S_3 + C_3\,S_1\,S_2)}$$

$$a_z = \mathrm{C_4\,(C_2\,C_3 - S_2\,S_3) + S_4\,(C_2\,S_3 + C_3\,S_2)}$$

$$p_x = l_2\,C_1\,C_2 - w\left(C_4\,(C_1\,C_2\,S_3 + C_1\,C_3\,S_2) - S_4\,(C_1\,C_2\,C_3 - C_1\,S_2\,S_3)\right) + C_5\,h\,(C_4\,(C_1\,C_2\,C_3 - C_1\,S_2\,S_3) - S_4\,(C_1\,C_2\,S_3 + C_1\,C_3\,S_2)) + l_3\,C_1\,C_2\,C_3 - l_3\,C_1\,S_2\,S_3 + S_1\,S_5\,h$$

$$p_y = l_2\,C_2\,S_1 - w\,(C_4\,(C_2\,S_1\,S_3 + C_3\,S_1\,S_2) - S_4\,(C_2\,C_3\,S_1 - S_1\,S_2\,S_3)) + C_5\,h\,(C_4\,(C_2\,C_3\,S_1 - S_1\,S_2\,S_3) - S_4\,(C_2\,S_1\,S_3 + C_3\,S_1\,S_2)) + l_3\,C_2\,C_3\,S_1 - C_1\,S_5\,h - l_3\,S_1\,S_2\,S_3$$

$$p_z = l_2\,S_2 + l_3 C_2\,S_3 + l_3\,C_3\,S_2 + w\,(C_4\,(C_2\,C_3 - S_2\,S_3) + S_4\,(C_2\,S_3 + C_3\,S_2)) + C_5\,h\,(C_4\,(C_2\,S_3 + C_3\,S_2) + S_4\,(C_2\,C_3 - S_2\,S_3))$$

- **Cálculo de la cinemática inversa**

Para obtener la cinemática inversa se ha realizado un desacoplo cinemático entre la parte del brazo y la de la muñeca y la mano. De este modo, partiendo de la posición deseada del efector final (p_x, p_y, p_z) y del ángulo θ_5 de la muñeca, se ha obtenido la posición $(p'_x,\ p'_y,\ p'_z)$ del sistema de referencia S_4. Finalmente utilizando estas nuevas coordenadas, se obtiene la cinemática inversa de la parte del brazo.

En primer lugar, se calcula el ángulo θ_1 partiendo de las coordenadas p_x y p_y del efector final según la ecuación (31) (ver Figura 98).

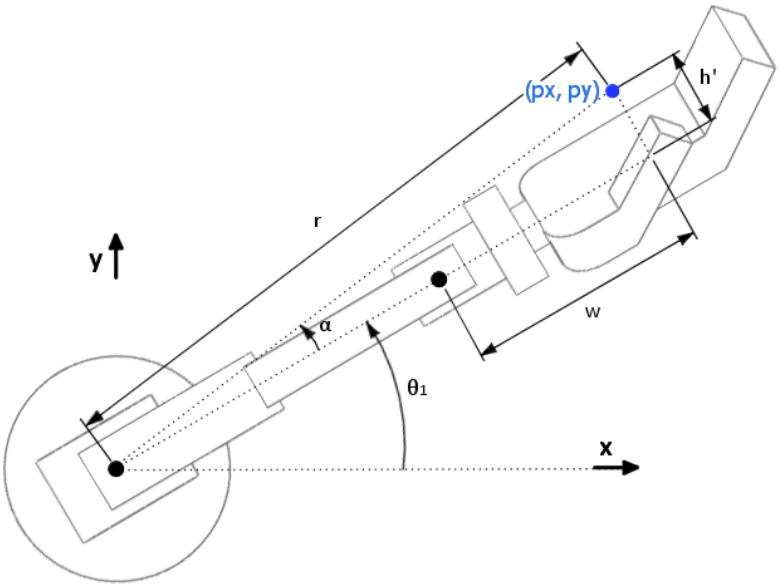


Figura 98. Obtención de θ_1

$$\theta_1 = arctan\left(\frac{p_y}{p_x}\right) - \alpha \quad \textbf{(31)}$$

Para obtener el ángulo α, es necesario obtener las distancias r y h'. Aplicando el teorema de Pitágoras (ecuación (32)) se obtiene la distancia r.

$$r = \sqrt{p_x^2 + p_y^2} \quad \textbf{(32)}$$

h' corresponde a la proyección horizontal de la distancia h (ecuación (33)). A su vez, h es la posición en el eje x del efector final en el sistema de referencia S_4 (ver Figura 99).

$$h' = h \cdot sen\ \theta_5 \quad \textbf{(33)}$$

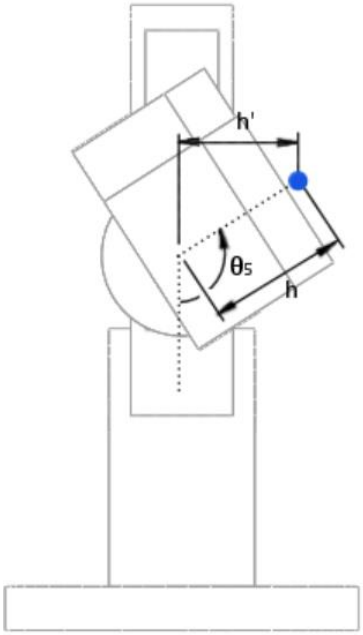


Figura 99. Obtención de θ_5

Una vez se conocen r y h', se obtiene α aplicando el arcoseno (ecuación (34)).

$$\alpha = \operatorname{arcsen}\left(\frac{h'}{r}\right) \quad \textbf{(34)}$$

Finalmente, sustituyendo (34) en (31)**(20)** se obtiene la ecuación (35).

$$\theta_1 = \arctan\left(\frac{p_y}{p_x}\right) - \operatorname{arcsen}\left(\frac{h'}{r}\right) \quad \textbf{(35)}$$

Y quedando θ_1 únicamente en funcion de los datos de partida queda la ecuación (36):

$$\theta_1 = \arctan\left(\frac{p_y}{p_x}\right) - \operatorname{arcsen}\left(\frac{h \cdot sen\ \theta_5}{\sqrt{p_x^2 + p_y^2}}\right) \quad \textbf{(36)}$$

Obtenido θ_1, puede calcularse la posición del sistema de referencia S_4 mediante las ecuaciones (37), (38) y (39).

$$p'_x = p_x - \Delta x \quad \textbf{(37)}$$

$$p'_y = p_y - \Delta y \quad \textbf{(38)}$$

$$p'_z = p_z - \Delta z \quad \textbf{(39)}$$

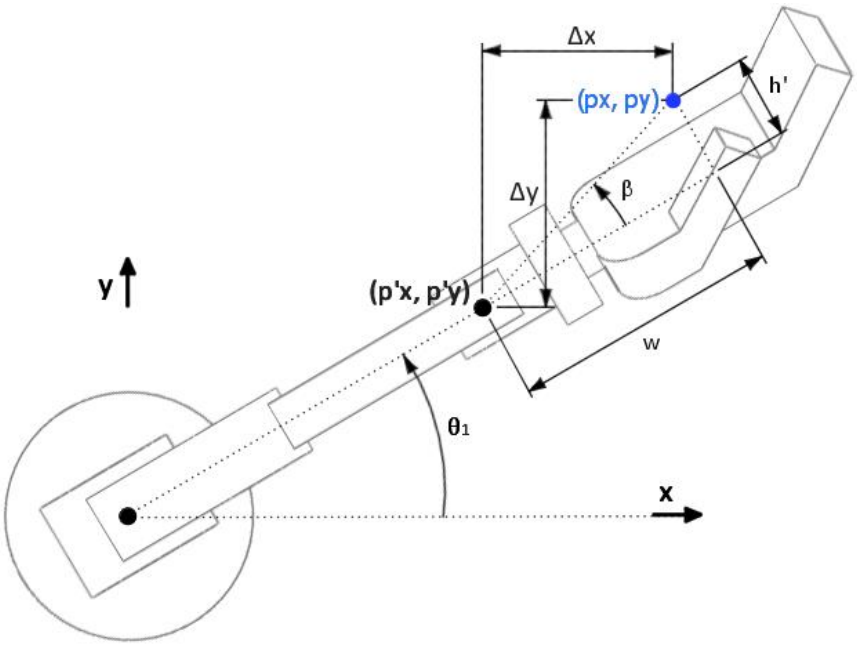


Figura 100. Obtención de las coordenadas (p'_x, p'_y)

Donde Δx e Δy se obtienen según las ecuaciones (40) y (41).

$$\Delta x = \sqrt{h'^2 + w^2} \cdot cos(\beta + \theta_1) \quad \textbf{(40)}$$

$$\Delta y = \sqrt{h'^2 + w^2} \cdot \text{sen}(\beta + \theta_1) \quad \textbf{(41)}$$

Siendo el ángulo β:

$$\beta = \arctan\left(\frac{h'}{w}\right) \quad \textbf{(42)}$$

w es la posición en el eje z del efector final en el sistema de referencia S_4.

Δz se obtiene como la proyección vertical de h según la ecuación (43) (ver Figura 101).

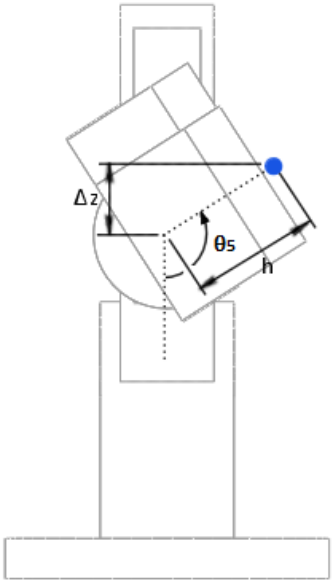


Figura 101. Obtención de θ_5

$$\Delta z = -h \cdot \cos\theta_5 \quad \textbf{(43)}$$

Partiendo de la nueva posición $(p'_x,\ p'_y,\ p'_z)$ se pueden obtener el resto de las ecuaciones para cinemática inversa. En primer lugar, se obtiene el ángulo θ_3 aplicando el teorema del coseno según las ecuaciones (44) y (45).

$$s^2 = l_2^2 + l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(180 + \theta_3) \rightarrow s^2 = l_2^2 + l_3^2 + 2 \cdot l_2 \cdot l_3 \cdot \cos\theta_3 \quad \textbf{(44)}$$

$$\cos\theta_3 = \frac{s^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3} \rightarrow\ \theta_3 = \arccos\left(\frac{s^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3}\right) \quad \textbf{(45)}$$

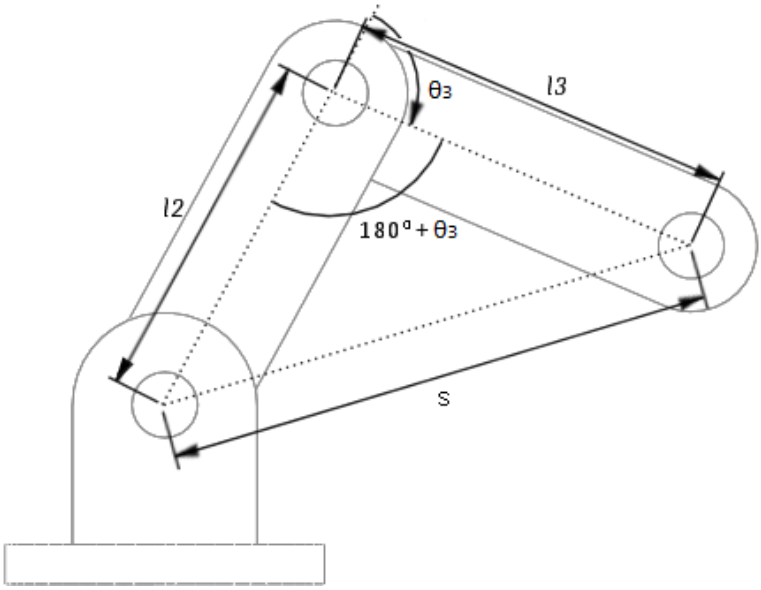


Figura 102. Obtención de θ_3

Por otro lado, se tiene que:

$$s^2 = p_x'^2 + p_y'^2 + p_z'^2 \quad \textbf{(46)}$$

Y sustituyendo (46) en (45) para quedar θ_3 en funcion de los datos de partida, se obtiene la ecuación (47):

$$\theta_3 = \arccos\left(\frac{p_x'^2 + p_y'^2 + p_z'^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3}\right) \quad \textbf{(47)}$$

Conocido θ_3 puede obtenerse el ángulo θ_2 según la ecuación (48) (ver Figura 103).

$$\theta_2 = \gamma + \delta \quad \textbf{(48)}$$

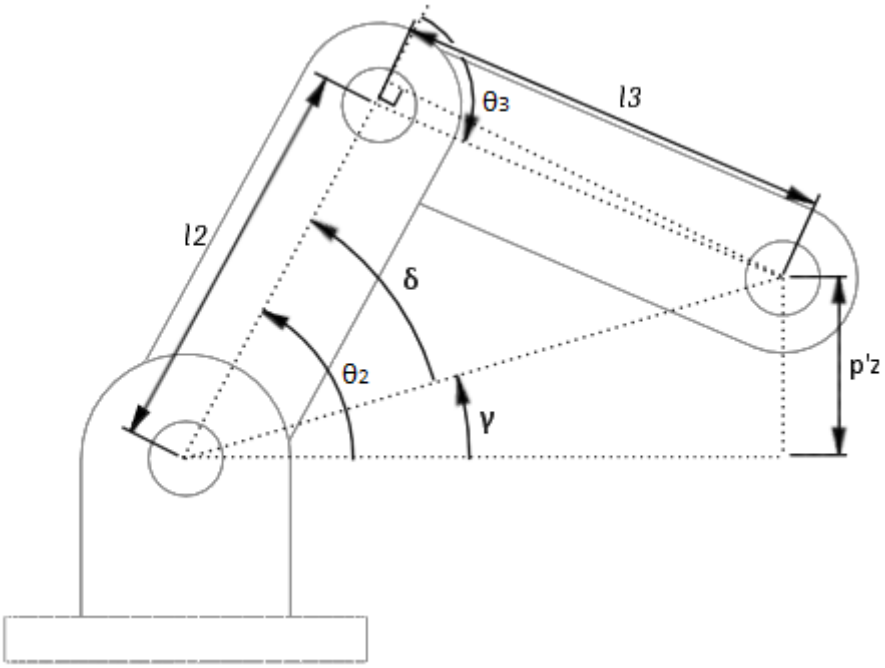


Figura 103. Obtención de θ_2

Donde γ se obtiene según la ecuación (49):

$$\gamma = \arctan\left(\frac{p'_z}{\sqrt{{p'_x}^2 + {p'_y}^2}}\right) \quad \textbf{(49)}$$

Y δ según la ecuación (50):

$$\delta = \arctan\left(-\frac{l_3 \cdot sen\,\theta_3}{l_2 + l_3 \cdot \cos\theta_3}\right) \quad \textbf{(50)}$$

Y sustituyendo (49) y (50) en (48) se obtiene la ecuación (51).

$$\theta_2 = \arctan\left(\frac{p'_z}{\sqrt{{p'_x}^2 + {p'_y}^2}}\right) + \arctan\left(\frac{l_3 \cdot sen\,\theta_3}{l_2 + l_3 \cdot \cos\theta_3}\right) \quad \textbf{(51)}$$

θ_4 está determinado por las relaciones mecánicas existentes en el robot según la ecuación (52).

$$\theta_4 = -(\theta_2 + \theta_3) \quad \textbf{(52)}$$

Finalmente, las expresiones de (36), (47) y (51) corresponden con la solución del problema cinemático inverso del brazo robot. Estas expresiones se recogen a continuación.

$$\theta_1 = \arctan\left(\frac{p_y}{p_x}\right) - \operatorname{arcsen}\left(\frac{h \cdot sen\,\theta_5}{\sqrt{p_x^2 + p_y^2}}\right)$$

$$\theta_2 = \arctan\left(\frac{p'_z}{\sqrt{{p'_x}^2 + {p'_y}^2}}\right) + \arctan\left(\frac{l_3 \cdot sen\,\theta_3}{l_2 + l_3 \cdot \cos\theta_3}\right)$$

$$\theta_3 = \arccos\left(\frac{p_x'^2 + p_y'^2 + p_z'^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3}\right)$$

No obstante, el fabricante utiliza un sistema de referencia distinto para definir los ángulos de las distintas articulaciones. Estos ángulos se obtienen a partir de los que se resultan de la cinemática inversa ($\theta_1, \theta_2\ y\ \theta_3$) según las ecuaciones (53), (54) y (55).

$$j_1 = \theta_1 \quad \textbf{(53)}$$

$$j_2 = 90° - \theta_2 \quad \textbf{(54)}$$

$$j_3 = -(\theta_2 + \theta_3) \quad \textbf{(55)}$$

- **Cálculo de velocidades**

Además de conocer la relación entre las coordenadas articulares y las del efector final, es interesante disponer de la relación entre sus velocidades para poder conseguir que el extremo desarrolle una trayectoria temporal concreta. Esta relación se puede obtener a través de la matriz jacobiana J según la ecuación (56).

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = J \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{bmatrix} \quad \textbf{(56)}$$

Siendo J:

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \cdots & \frac{\partial p_x}{\partial q_5} \\ \vdots & \ddots & \vdots \\ \frac{\partial \gamma}{\partial q_1} & \cdots & \frac{\partial \gamma}{\partial q_5} \end{bmatrix}$$

En este caso, no existen grados de libertad de suficientes para considerar las velocidades angulares de orientación del efector final, por lo que se eliminan las tres últimas filas de la matriz. q_4 se puede obtener como combinación linear del resto de variables y por tanto se pueden eliminar los elementos. Finalmente, se tiene la ecuación (31).

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = J \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_5 \end{bmatrix} \quad \textbf{(57)}$$

Siendo la matriz jacobiana:

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} & \frac{\partial p_x}{\partial q_5} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} & \frac{\partial p_y}{\partial q_5} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} & \frac{\partial p_z}{\partial q_5} \end{bmatrix}$$

Para poder calcular la matriz jacobiana primero es necesario obtener la cinemática directa del robot, la cual se define mediante las ecuaciones (58), (59) y (60).

$$p_x = (l_2 \cdot \cos q_2 + l_3 \cdot \cos(q_2 + q_3) + w) \cdot \cos q_1 - h \cdot \operatorname{sen} q_5 \cdot \operatorname{sen} q_1 \quad \textbf{(58)}$$

$$p_y = (l_2 \cdot \cos q_2 + l_3 \cdot \cos(q_2 + q_3)) \cdot \operatorname{sen} q_1 + h \cdot \operatorname{sen} q_5 \cdot \cos q_1 \quad \textbf{(59)}$$

$$p_z = l_2 \cdot sen\ q_2 + l_3 \cdot sen\ (q_2 + q_3) - h \cdot \cos q_5 \quad \textbf{(60)}$$

Una vez obtenidas las ecuaciones que relacionan las coordenadas del efector final con las coordenadas articulares se calculan las derivadas parciales que forman la matriz jacobiana. A continuación, se desarrollan los términos de la matriz jacobiana:

$$\frac{\partial p_x}{\partial q_1} = -(l_2 \cdot \cos q_2 + l_3 \cdot \cos(q_2 + q_3) + w) \cdot sen\ q_1 - h \cdot \operatorname{sen} q_5 \cdot \cos q_1$$

$$\frac{\partial p_x}{\partial q_2} = \left(-l_2 \cdot sen\ q_2 - l_3 \cdot sen(q_2 + q_3)\right) \cdot \cos q_1$$

$$\frac{\partial p_x}{\partial q_3} = -l_3 \cdot sen\ (q_2 + q_3) \cdot \cos q_1$$

$$\frac{\partial p_x}{\partial q_5} = -\cos q_5 \cdot \operatorname{sen} q_1$$

$$\frac{\partial p_y}{\partial q_1} = (l_2 \cdot \cos q_2 + l_3 \cdot \cos(q_2 + q_3)) \cdot \cos q_1 - h \cdot \sin q_5 \cdot \cos q_1$$

$$\frac{\partial p_y}{\partial q_2} = \left(-l_2 \cdot sen\ q_2 - l_3 \cdot sen(q_2 + q_3)\right) \cdot \operatorname{sen} q_1$$

$$\frac{\partial p_y}{\partial q_3} = -l_3 \cdot sen\,(q_2 + q_3) \cdot \text{sen}\, q_1$$

$$\frac{\partial p_y}{\partial q_5} = h \cdot \cos q_5 \cdot \cos q_1$$

$$\frac{\partial p_z}{\partial q_1} = 0$$

$$\frac{\partial p_z}{\partial q_2} = l_2 \cdot \cos q_2 + l_3 \cdot \cos(q_2 + q_3)$$

$$\frac{\partial p_z}{\partial q_3} = l_3 \cdot \cos(q_2 + q_3)$$

$$\frac{\partial p_z}{\partial q_5} = h \cdot \text{sen}\, q_5$$

ANEXO IV: CÓDIGO DE MATLAB

A continuación, se explican todas las funciones y código de *MATLAB* que se ha elaborado para el control del robot. Se pueden diferenciar dos grupos, las funciones elaboradas para facilitar el uso de la API del fabricante y las funciones para realizar el algoritmo de control.

- **API propia**

El fabricante proporciona una API en forma de librería dinámica. Se ha creado una capa de abstracción o API propia para evitar tener que tratar continuamente con llamadas a esta librería y evitar el uso de punteros necesarios para hacer dichas llamadas. De esta forma, se consigue poder utilizar las funciones proporcionadas por el fabricante como cualquier otra función típica de *MATLAB*.

- **MgLoadDll**

```
function MgLoadDll()
%MGLOADDLL Carga la librería dinámica DobotDll.dll
%   Habilita el uso de las funciones para el control del
%   brazo robot Dobot Magician
    if ~libisloaded('DobotDll')
        [~,~] = loadlibrary('DobotDll','DobotDll.h');
    else
        fprintf('La librería ya está cargada\n');
    end
end
```

- **MgUnloadDll**

```
function MgUnloadDll()
%MGLOADDLL Carga la librería dinámica DobotDll.dll
%   Habilita el uso de las funciones para el control del
%   brazo robot Dobot Magician
    if libisloaded('DobotDll')
        unloadlibrary('DobotDll');
    else
        fprintf('La librería no está cargada\n');
    end
end
```

- **MgSearchDobot**

```
function addr = MgSearchDobot()
%MGSEARCHDOBOT Busca el dispositivo Dobot y devuelve el puerto COM al que
%está conectado (addr)
    ch = blanks(128);
    ch_cptr = libpointer('cstring',ch);
    [~, addr] = calllib('DobotDll','SearchDobot',ch_cptr,128);
    if isempty(addr)
        fprintf('No se ha encontrado ningun dispositivo\n');
    end
end
```

- **MgConnectDobot**

```
function res = MgConnectDobot(addr)
%MGCONNECTDOBOT(addr) Establece la conexión con el brazo robot
%   'addr': puerto COM al que está conectado el robot en formato string
%
%   El valor de 'res' dependerá del resultado de la conexión
%   res = 0 -> Conexión establecida
%   res = 1 -> Error de conexión
%   res = 2 -> Tiempo de conexión agotado
    baudrate = 115200;
    if ~ischar(addr)
        error(message('addr debe ser de tipo string\n'));
    end

    addr_cptr = libpointer('cstring',addr);

    [res,~]=calllib('DobotDll','ConnectDobot',addr_cptr, baudrate);

    switch res
        case 0
            fprintf('Conexión establecida (%s)\n',addr);
        case 1
            fprintf('Error de conexión\n');
        case 2
            fprintf('Tiempo de conexión agotado\n');
    end
end
```

- **MgDisconnectDobot**

```
function MgDisconnectDobot()
%MGDISCONNECTDOBOT Desconecta el dispositivo dobot
    calllib('DobotDll','DisconnectDobot');
end
```

- **MgSetPTPCmd**

```
function index = MgSetPTPCmd(cmd)
%MGSETPTPCMD Añade un comando PTP (punto a punto) a la cola de comandos
%   cmd debe ser un objeto de tipo MgPTPCmd
%
%   Devuelve la posicion del comando en la cola.
    index = 0;
    index_ptr = libpointer('uint64Ptr',index);
    cmd_ptr = cmd.StructPointer();
    [~,~,index] = calllib('DobotDll','SetPTPCmd',cmd_ptr,true,index_ptr);
end
```

- **MgPTPCmd**

```
classdef MgPTPCmd
    %MGPTPCMD Parámetros para comando PTP (punto a punto)
    properties
        ptpMode
        x
        y
        z
```

```
        r
    end
    methods
        function obj = MgPTPCmd(ptpMode,x,y,z,r)
            %MGPTPCMD Constructor de la clase
            obj.ptpMode = ptpMode;
            obj.x = x;
            obj.y = y;
            obj.z = z;
            obj.r = r;
        end
        function struct_ptr = StructPointer(PTPCmd)
            %STRUCTPOINTER devuelve un puntero a la estructura de datos
            mtlb_struct.ptpMode = uint8(PTPCmd.ptpMode);
            mtlb_struct.x = PTPCmd.x;
            mtlb_struct.y = PTPCmd.y;
            mtlb_struct.z = PTPCmd.z;
            mtlb_struct.r = PTPCmd.r;
            c_struct = libstruct('tagPTPCmd',mtlb_struct);
            struct_ptr = libpointer('tagPTPCmdPtr',c_struct);
        end
    end
end
```

- **MgPTPMode**

```
% Enumeración de los modos PTP (punto a punto)
classdef MgPTPMode < uint8
    enumeration
        JUMP_XYZ(0),
        MOVJ_XYZ(1),
        MOVL_XYZ(2),
        JUMP_ANGLE(3),
        MOVJ_ANGLE(4),
        MOVL_ANGLE(5),
        MOVJ_INC(6),
        MOVL_INC(7),
        MOVJ_XYZ_INC(8),
        JUMP_MOVL_XYZ(9)
    end
end
```

- **MgSetQueuedCmdStartExec**

```
function MgSetQueuedCmdStartExec()
%MGSETQUEUEDCMDSTARTEXEC Da comienzo a la ejecución de comandos en la cola
    calllib('DobotDll','SetQueuedCmdStartExec');
end
```

- **MgSetQueuedCmdStopExec**

```
function MgSetQueuedCmdStopExec()
%MGSETQUEUEDCMDSTOPEXEC Detiene la ejecución de los comandos en cola
%   Esta funcion no fuerza la parada del comando que ya esté en ejecución
    calllib('DobotDll','SetQueuedCmdStopExec');
end
```

- **MgGetQueuedCmdCurrentIndex**

```
function index = MgGetQueuedCmdCurrentIndex()
%MGGETQUEUEDCMDCURRENTINDEX Devuelve la indice en la cola del comando en
%ejecución
    index = 0;
    index_ptr = libpointer('uint64Ptr',index);
    [~,index] = calllib('DobotDll','GetQueuedCmdCurrentIndex',index_ptr);
end
```

- **MgSetWAITCmd**

```
function index = MgSetWAITCmd(time_ms)
%MGSETWAITCMD Añade un comando de espera a la cola
%   time_ms es el tiempo que el robot permanecerá quieto en milisegundos.
%
%   Devuelve la posicion del comando en la cola.
    index = 0;
    index_ptr = libpointer('uint64Ptr',index);
    mtlb_struct.timeout = time_ms;
    c_struct = libstruct('tagWAITCmd',mtlb_struct);
    struct_ptr = libpointer('tagWAITCmdPtr',c_struct);
    [~,~,index] = calllib('DobotDll','SetWAITCmd',struct_ptr,true,index_ptr);
end
```

- **Funciones para el algoritmo de control**

Por una parte, ha sido necesaria la definición del modelo cinemático del robot dentro de *MATLAB*. Para ello se ha utilizado las herramientas proporcionadas por la *Robotics System Toolbox*.

Para el cálculo de la cinemática directa dicha librería ofrece métodos basados en la definición del modelo cinemático mediante matrices de transformación homogénea.

Además, también ofrece varios métodos para el cálculo de la cinemática inversa. Estos métodos están basados en un proceso iterativo basado en condiciones y restricciones. Esto permite calcular la cinemática inversa para situaciones de cierta complejidad o para robots con multitud de grados de libertad, no obstante, tienen un coste computacional elevado. En el caso del brazo robot *Dobot*, la cinemática es sencilla y se contemplan casos sencillos. Por este motivo, se han obtenido las ecuaciones que definen la cinemática inversa, que permiten obtener un resultado directo y mucho más rápido.

Por otro lado, también se han elaborado funciones para facilitar la programación de los movimientos del robot.

- **LoadDobotModel**

```
function dobot = LoadDobotModel()
    %LOADDOBOTMODEL Devuelve el árbol de objetos que define el robot Dobot
    %Magician junto con la mano protésica.
    h = 10; %mm
    w = 220; %mm
    l2 = 135; %mm
    l3 = 147; %mm
```

```
dobot = robotics.RigidBodyTree();

addVisual(dobot.Base,'Mesh','DobotMeshes/base.stl');

shoulder = robotics.RigidBody('shoulder');
j1 = robotics.Joint('q1','revolute');
j1.JointAxis = [0, 0, 1]; %Eje de rotación respecto al stma de ref del padre
tform = trvec2tform([0,0,-0.082]); %Traslación respecto al padre
setFixedTransform(j1,tform);
j1.HomePosition = 0;
j1.PositionLimits = [-pi/2,pi/2];
shoulder.Joint = j1;
addVisual(shoulder,'Mesh','DobotMeshes/hombro.stl');
addBody(dobot, shoulder, 'base');

arm = robotics.RigidBody('arm');
j2 = robotics.Joint('q2','revolute');
ttform = trvec2tform([0,0,0.082]); %Traslación respecto al padre
rxtform = eul2tform([0,0, pi/2]); %[z, y, x] Giro en X para colocar eje Z
rztform = eul2tform([0,0, 0]); %[z, y, x] Giro en Z para colocar eje X
transform = ttform * rxtform * rztform;
setFixedTransform(j2,transform);
j2.JointAxis = [0, 0, 1]; %Eje de rotación stma de ref local
j2.HomePosition = pi/4;
j2.PositionLimits = [0,deg2rad(85)];
arm.Joint = j2;
addVisual(arm,'Mesh','DobotMeshes/brazo.stl',rxtform');
addBody(dobot, arm, 'shoulder');

forearm = robotics.RigidBody('forearm');
j3 = robotics.Joint('q3','revolute');
ttform = trvec2tform([l2/1000,0,0]); %Traslación respecto al padre
transform = ttform;
setFixedTransform(j3,transform);
j3.JointAxis = [0, 0, 1]; %Eje de rotación
j3.HomePosition = -pi/2;
j3.PositionLimits = [-deg2rad(145),-deg2rad(10)];
forearm.Joint = j3;
addVisual(forearm,'Mesh','DobotMeshes/antebrazo.stl',rxtform');
addBody(dobot, forearm, 'arm');

wrist = robotics.RigidBody('wrist');
j4 = robotics.Joint('q4','revolute');
ttform = trvec2tform([l3/1000,0,0]); %Traslación respecto al padre
transform = ttform;
setFixedTransform(j4,transform);
j4.JointAxis = [0, 0, 1]; %Eje de rotación
j4.HomePosition = pi/4;
j4.PositionLimits = [-pi/4, pi/2];
wrist.Joint = j4;
addVisual(wrist,'Mesh','DobotMeshes/muneca.stl');
addBody(dobot, wrist, 'forearm');

hand = robotics.RigidBody('hand');
j5 = robotics.Joint('q5','revolute');
ttform = trvec2tform([0,0,0]); %Traslación respecto al padre
rztform = eul2tform([-pi/2,0, 0]); %[z, y, x] Giro en Z para colocar eje X
rxtform2 = eul2tform([0,0, -pi/2]);
```

```
    transform = ttform * rztform * rxtform2;
    setFixedTransform(j5,transform);
    j5.JointAxis = [0, 0, 1]; %Eje de rotación
    j5.HomePosition = 0;
    j5.PositionLimits = [0,pi];
    hand.Joint = j5;
    mesh_tform = trvec2tform([0,0,0.108]);
    addVisual(hand,'Mesh','DobotMeshes/mano.stl', mesh_tform);
    addBody(dobot, hand, 'wrist');

    effector = robotics.RigidBody('effector');
    j6 = robotics.Joint('effector','fixed');
    ttform = trvec2tform([h/1000,0,w/1000]); %Traslación respecto al padre
    transform = ttform;
    setFixedTransform(j6,transform);
    effector.Joint = j6;
    addBody(dobot, effector, 'hand');
end
```

- **DirectKinem**

```
function coords = DirectKinem(robot, pose)
%DIRECTKINEM Devuelve la posicion en mm del efector final dado el ángulo de
%cada articulación
    coords = 1000*tform2trvec(getTransform(robot,pose,'effector'));
end
```

- **InvKinem**

```
function pose = InvKinem(robot,coords,q5)
%Dado un punto destino para el efector final y un ángulo para la rotación
%de la muñeca, devuelve la configuración del robot definida por los ángulos
%de cada articulación (q1,q2,q3,q4,q5) necesaria para llevar el efector
%final al punto especificado.

    px = coords(1);
    py = coords(2);
    pz = coords(3);

    l2 = 135; %mm
    l3 = 147; %mm

    effector_offsets =
tform2trvec(getTransform(robot,robot.homeConfiguration,'effector'))...
        - tform2trvec(getTransform(robot,robot.homeConfiguration,'hand'));
    h = sqrt((effector_offsets(2)*1000)^2+(effector_offsets(3)*1000)^2);
    w = effector_offsets(1)*1000;

    r = sqrt(px^2+py^2);
    h_p = h * sin(q5);

    alpha = atan((h_p/r)/sqrt(1-(h_p/r)^2));
    q1 = atan(py/px)- alpha;

    beta = atan(h_p/w);
    delta_x = sqrt(h_p^2+w^2)*cos(beta+q1);
    delta_y = sqrt(h_p^2+w^2)*sin(beta+q1);
```

```
    delta_z = -h*cos(q5);

    px_p = px-delta_x;
    py_p = py-delta_y;
    pz_p = pz-delta_z;

    s2 = px_p^2+py_p^2+pz_p^2;
    cos_q3 = (s2-l2^2-l3^2)/(2*l2*l3);
    q3 = -acos(cos_q3);

    gamma = atan(pz_p/sqrt(px_p^2+py_p^2));
    delta = -atan(l3*sin(q3)/abs(l2+l3*cos_q3));

    q2 = gamma + delta;

    q4 = -(q2+q3);

    pose(1).JointName = 'q1';
    pose(1).JointPosition = q1;
    pose(2).JointName = 'q2';
    pose(2).JointPosition = q2;
    pose(3).JointName = 'q3';
    pose(3).JointPosition = q3;
    pose(4).JointName = 'q4';
    pose(4).JointPosition = q4;
    pose(5).JointName = 'q5';
    pose(5).JointPosition = q5;

end
```

- **AddTrayectoryToQueue**

```
function last_index = AddTrayectoryToQueue(poses)
%MOVEDOBOT Añade una secuencia de comandos de movimiento a la cola
%definidos por la lista de posturas 'poses'.
%
%poses debe ser una matriz donde cada fila corresponde con una postura.
%
%Devuelve la posición en la cola del último comando añadido.
    for i = 1:size(poses,1)
        point = poses(i,:);
        j1 = rad2deg(point(1).JointPosition);
        q2 = rad2deg(point(2).JointPosition);
        j2 = 90 - q2;
        q3 = rad2deg(point(3).JointPosition);
        j3 = -(q2+q3);
        r = rad2deg(point(5).JointPosition);
        comand = MgPTPCmd(MgPTPMode.MOVJ_ANGLE, j1, j2, j3, r);
        last_index = MgSetPTPCmd(comand);
    end
end
```

- **MoveDobot**

```
function frames = MoveDobot(waypoints, simulated)
%MOVEDOBOT Comando para mover el robot.
%
```

```
%simulated = 0: Se mandan los comandos al robot para que realice el
%movimiento.
%simulated = 1: Se muestra la simulación de la trayectoria sin mandar
%ningún comando al robot.

    dobot = LoadDobotModel();

    points(size(waypoints,1),5) = struct('JointName',[],'JointPosition',[]);
    for i = 1:size(waypoints,1)
        points(i,:) = InvKinem(dobot,waypoints(i,1:3),waypoints(i,4));
        points(i,1).JointName = 'q1';
        points(i,2).JointName = 'q2';
        points(i,3).JointName = 'q3';
        points(i,4).JointName = 'q4';
        points(i,5).JointName = 'q5';
        for j = 1:size(points,2)
            limits = dobot.Bodies{j}.Joint.PositionLimits;
            position = points(i,j).JointPosition;
            if limits(1)>position || limits(2)<position
                error('Articulación %s fuera de rango (%.2f<%.2f<%.2f). Punto
[%.2f %.2f %.2f
%.2f]',points(i,j).JointName,limits(1),position,limits(2),waypoints(i,1),waypoin
ts(i,2),waypoints(i,3),waypoints(i,4));
            end
        end
    end

    if simulated == 0
        MgLoadDll();
        addr = MgSearchDobot();
        if isempty(addr)
            return;
        end
        if MgConnectDobot(addr) ~= 0
            return;

        end

        last_index = AddTrayectoryToQueue(points, interp_mode);

        MgSetQueuedCmdStartExec();

        while MgGetQueuedCmdCurrentIndex() < last_index
          pause(0.5);
        end

        MgSetQueuedCmdStopExec();
        frames = 0;
    else
        framerate = 15; %Imágenes por segundo (fps) en la simulación
        tFinal = 10; %Tiempo total de la simulación en segundos
        tWaypoints = [0,linspace(2,tFinal,size(waypoints,1)-1)];
        numFrames = tFinal*framerate;

        points_coord = zeros(size(points));
        for i = 1:size(points,1)
            for j = 1:size(points,2)
```

```
                points_coord(i,j) = points(i,j).JointPosition;
            end
        end

        if size(points_coord,1) == 1
            numFrames = 1;
            points_interp = points_coord;
        else
            points_interp =
interp1(tWaypoints,points_coord,linspace(0,tFinal,numFrames));


            for i = 1:size(points_interp,1)
                points_interp(i,4) = -(points_interp(i,2)+points_interp(i,3));
            end
        end


        pose_interp(size(points_interp,1),5) =
struct('JointName',[],'JointPosition',[]);
        for i = 1:size(points_interp,1)
            pose_interp(i,1).JointName = 'q1';
            pose_interp(i,2).JointName = 'q2';
            pose_interp(i,3).JointName = 'q3';
            pose_interp(i,4).JointName = 'q4';
            pose_interp(i,5).JointName = 'q5';
            for j = 1:size(points_interp,2)
                pose_interp(i,j).JointPosition = points_interp(i,j);
            end
        end

        endEffectorPosition = zeros(numFrames,3);
        for k = 1:numFrames
            endEffectorPosition(k,:) = DirectKinem(dobot,pose_interp(k,:))/1000;
        end

        figure('Units','normalized','Position',[0.1,0.1,0.75,0.75])
        h = show(dobot, pose_interp(1,:),'PreservePlot',false,'Frames','off');
        set(h,'Position',[0,0,1,1])
        xlim([-0.2,0.7])
        ylim([-0.6,0.6])
        zlim([-0.2,0.3])
        xlabel('Eje X (m)');
        ylabel('Eje Y (m)');
        zlabel('Eje Z (m)');
        title('Simulación de movimiento');
        hold on
        p =
plot3(endEffectorPosition(1,1),endEffectorPosition(1,2),endEffectorPosition(1,3)
,'b');

        [x, y] = meshgrid(-1:0.1:1); % Generar datos de x e y
        z = ones(size(x)).*-0.140; % Generar datos de z
        floor = surf(x, y, z,'FaceColor',[0.75 0.6
0.25],'EdgeColor',[.4,.4,.4]); % Trazar la superficie
        alpha(floor,0.7);
        f(numFrames) = struct('cdata',[],'colormap',[]);
```

```
        for k = 1:size(pose_interp,1)
            h = show(dobot,
pose_interp(k,:),'PreservePlot',false,'Frames','off');
            set(h,'Position',[0,0,1,1])
            xlim([-0.2,0.7])
            ylim([-0.6,0.6])
            zlim([-0.2,0.3])
            p.XData(k)=endEffectorPosition(k,1);
            p.YData(k)=endEffectorPosition(k,2);
            p.ZData(k)=endEffectorPosition(k,3);
            f(k) = getframe;
        end
        hold off
        frames = f;
    end
end
```

- **GetJacobian**

```
function J = GetJacobian(q1,q2,q3,q5)
%GETJACOBIAN Returns the jacobian matrix for the position (q1,q2,q3,q5).
    h = 10; %mm
    w = 220; %mm
    l2 = 135; %mm
    l3 = 147; %mm

    dpx_dq1 = -(l2*cos(q2)+l3*cos(q2+q3)+w)*sin(q1)-h*sin(q5)*cos(q1);
    dpx_dq2 = (-l2*sin(q2)-l3*sin(q2+q3))*cos(q1);
    dpx_dq3 = -l3*sin(q2+q3)*cos(q1);
    dpx_dq5 = -h*cos(q5)*sin(q1);

    dpy_dq1 = (l2*cos(q2)+l3*cos(q2+q3)+w)*cos(q1)-h*sin(q5)*sin(q1);
    dpy_dq2 = (-l2*sin(q2)-l3*sin(q2+q3))*sin(q1);
    dpy_dq3 = -l3*sin(q2+q3)*sin(q1);
    dpy_dq5 = h*cos(q5)*cos(q1);

    dpz_dq1 = 0;
    dpz_dq2 = l2*cos(q2)+l3*cos(q2+q3);
    dpz_dq3 = l3*cos(q2+q3);
    dpz_dq5 = h*sin(q5);

    J = [dpx_dq1, dpx_dq2, dpx_dq3, dpx_dq5;
        dpy_dq1, dpy_dq2, dpy_dq3, dpy_dq5;
        dpz_dq1, dpz_dq2, dpz_dq3, dpz_dq5;];
end
```

- **GetEndEffectorSpeeds**

```
function v = GetEndEffectorSpeeds(q1,q2,q3,q5,v_q1,v_q2,v_q3,v_q5)
%GETENDEFFECTORSPEEDS Devuelve la velocidad en mm/s [vx,vy,vz] del efector
%final para una posicion y velocidad especifica de las
%articulaciones.(q1,q2,q3,q5) deben estar en radianes.

    J = GetJacobian(q1,q2,q3,q5);

    v = J * [v_q1;v_q2;v_q3;v_q5];
end
```

- **GetJointSpeeds**

```
function v = GetJointSpeeds(q1,q2,q3,q5,vx,vy,vz)
%GETJOINTSPEEDS Devuelve la velocidad en rad/s de cada articulación para
%una posicion articular y una velocidad del efector final.(q1,q2,q3,q5)
%debe estar en radianes.

    J = GetJacobian(q1,q2,q3,q5);

    J_p = inv(J'*J)*J';

    v = J_p *[vx;vy;vz];

end
```

• Ejemplos de control

A continuación, se muestran varios ejemplos de simulaciones realizadas en el programa.

Para cambiar la trayectoria realizada por el robot basta con cambiar la variable que representa la trayectoria (ver apartado 6.7.7).

En el siguiente ejemplo, el efector final realiza una trayectoria trazando un cuadrado (ver Figura 104).

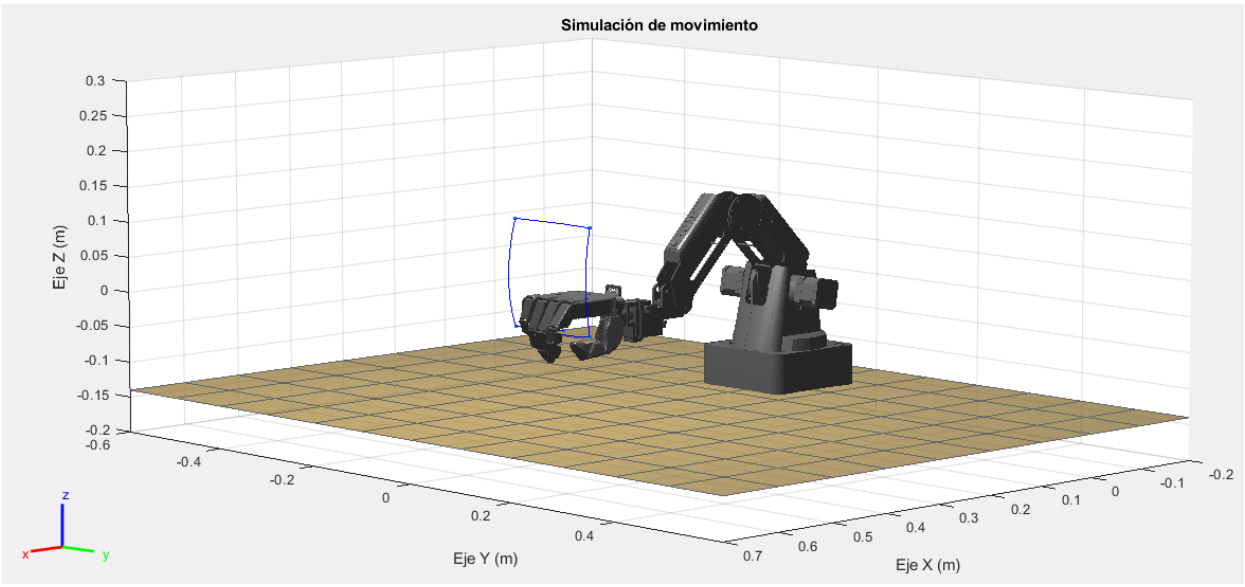


Figura 104. Ejemplo 2 del control

Los puntos se han definido de la siguiente forma:

```
waypoints = [home_pos,0;...
             home_pos(1)+20,home_pos(2),home_pos(3),0;...
             home_pos(1)+20,home_pos(2)+75,home_pos(3),pi/2;...
             home_pos(1)+20,home_pos(2)+75,home_pos(3)+150,pi;...
             home_pos(1)+20,home_pos(2)-75,home_pos(3)+150,pi/2;...
             home_pos(1)+20,home_pos(2)-75,home_pos(3),0;...
             home_pos(1)+20,home_pos(2),home_pos(3),0;...
```

```
            home_pos,0];
```

En el siguiente ejemplo mostrado en la Figura 105 el efector final realiza una trayectoria circular.

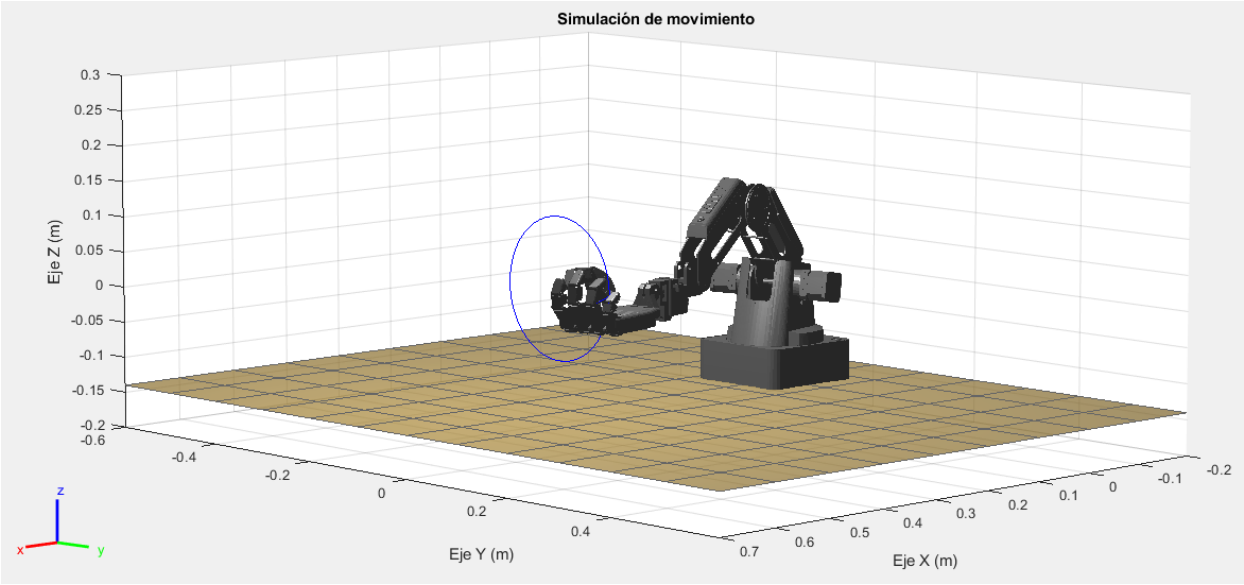


Figura 105. Ejemplo 3 del control

Los puntos se han definido de la siguiente forma:

```
n = 100; %Número de puntos de la trayectoria
r = 100; %Radio en mm
center = [home_pos(1), home_pos(2), home_pos(3)+50]; %Coordenadas del centro en
mm
start_angle = 0; %Ángulo de inicio
theta = linspace(start_angle,start_angle+2*pi,n);
x = ones(1,n)*center(1);
y = center(2) + r*cos(theta);
z = center(3) + r*sin(theta);
q5 = linspace(0,pi,n);
waypoints = [home_pos 0];
waypoints = [waypoints; x' y' z' q5'];
```

2. PLIEGO DE CONDICIONES TÉCNICAS

• Introducción

El presente pliego de condiciones técnicas tiene como misión establece las condiciones técnicas para que el proyecto pueda materializarse bajo las condiciones especificadas.

A continuación, se recogen todas aquellas especificaciones técnicas a tener en cuenta para la correcta materialización del proyecto.

• Especificaciones de los materiales

El material utilizado para fabricar las piezas será ácido poliláctico (PLA) en forma de filamento bobinado. El filamento deberá ser de calidad para asegurar que las piezas se fabricaran bajo las tolerancias especificadas y que tendrán la resistencia adecuada. Las propiedades más relevantes del PLA se muestran en la siguiente tabla. Los valores que aparecen en la tabla son los utilizados en los análisis y cálculos.

Densidad	1,25	kg/cm^3
Módulo de Young	3,5	GPa
Límite elástico	50	MPa
Coeficiente de Poisson	0,39	

El material utilizado para la tornillería necesaria para el montaje será el indicado por la norma aplicable a cada uno de los elementos. Los tornillos y tuercas utilizados pueden consultarse en el documento PLANOS.

• Especificaciones de los equipos

Serán necesarios los siguientes equipos:

- Brazo robótico *Dobot Magician*
- Ordenador. Deberá tener la potencia suficiente para ejecutar el software necesario para controlar el robot. Para el software usado en la fabricación de las piezas podrá utilizarse el mismo ordenador u otro destinado a tal uso.
- Impresora 3D. Deberá ser capaz de fabricar las piezas con las tolerancias especificadas.

• Especificaciones del software

Todo el software utilizado y sus respectivas licencias deberán ser adquiridos a través de los suministradores oficiales. Los programas necesarios se especifican a continuación.

- *MATLAB*. Se utilizará la versión 2019a o superior.
- *Robotics System Toolbox*. Complemento de *MATLAB*. Se deberá utilizar la versión 2.2. Se podrán utilizar versiones posteriores siempre y cuando estas mantengan la compatibilidad con la versión mencionada.

- Software para la impresión 3D ("*Slicer*"). Se podrá utilizar cualquier software que permita fabricar las piezas bajo las especificaciones de resistencia y tolerancia

• Especificaciones de fabricación

Las piezas se fabricarán según las indicaciones establecidas en el apartado 0 de la memoria y deberán cumplir con las dimensiones y tolerancias especificadas en el documento PLANOS.

• Especificaciones de montaje

Para realizar la modificación del robot se seguirán los pasos establecidos en el apartado 6.2 de la memoria. Además, se tomarán las precauciones de seguridad necesarias para evitar posibles lesiones durante la modificación.

• Instrucciones de uso

Para las instrucciones de uso relativas a la manipulación del brazo robot (encendido, apagado, mantenimiento, etc.) se seguirán las indicaciones proporcionadas por el fabricante[6]. De la misma forma, se respetarán las advertencias de seguridad establecidas por el fabricante.

Para controlar el posicionado del robot en base a puntos de paso, puede utilizarse la funcion *MoveDobot* según se explica en el apartado 6.7.7.

Se permitirá la modificación de las funciones elaboradas en este proyecto (ver Anexo IV: Código de MATLAB), no obstante no se asegura que al modificar una función el resto sigan funcionando correctamente o que se sigan comportando según lo dispuesto en los apartados 6.7.4 y 6.7.6. Alternativamente se recomienda crear funciones nuevas en caso de que se quiera modificar o ampliar el control del robot.

La carga máxima que se podrá colocar en el sistema de acople será de 0,5 Kg incluyendo el peso de la mano protésica.

3. PRESUPUESTO

- ## Introducción

En este apartado se detallan las estimaciones de todos los costes relativos a este proyecto. Se tendrán en cuenta los costes relativos al diseño de las piezas y su coste de fabricación, así como el coste de desarrollo del software y de todo el equipo necesario.

- ## Amortización de los equipos

Equipo	Precio (€)	Periodo de amortización (Años)	Tiempo de uso (Días)	Coste (€)
Ordenador de laboratorio	1500	5	25	20,55
Ordenador portátil	1000	5	125	68,49
Impresora 3D	300	2	3	1,23
			Total	90,27

- ## Material

A continuación, se detalla el coste de las piezas debido únicamente al material utilizado y el coste de la tornillería. La estimación del coste de las piezas se ha calculado a partir del precio de la bobina de PLA (22€) y de la estimación de material utilizado que proporciona el programa utilizado para la fabricación de las piezas puesto que tiene en cuenta también el material de soporte utilizado.

Descripción	Cantidad	Coste unitario (€)	Coste total (€)
Acoplador de la mano	1	0,46	0,46
Puente del servo	1	0,44	0,44
Soporte del servo	1	0,75	0,75
Soporte de los actuadores	1	0,97	0,97
Sujeción del soporte de los actuadores	2	0,13	0,26
Tornillo ISO 4762 M8x25	2	0,08	0,16
Tornillo ISO 4762 M5x25	1	0,07	0,07
Tuerca ISO 4035 M5	1	0,05	0,05
Tornillo ISO 4762 M4x25	2	0,07	0,14
Tornillo ISO 4762 M4x20	8	0,07	0,56
Tuerca ISO 4035 M4	10	0,05	0,5
Tornillo ISO 4762 M3x16	2	0,07	0,14
Tuerca ISO 4035 M3	2	0,05	0,1
		Total	4,6

- **Software**

Descripción	Precio/año (€)	Tiempo de uso (Días)	Coste (€)
Matlab + Robotics System Toolbox	1000	100	273,97
Solidworks	3000	40	328,77
Ms Office	69	90	17,01
		Total	615,97

- **Mano de obra**

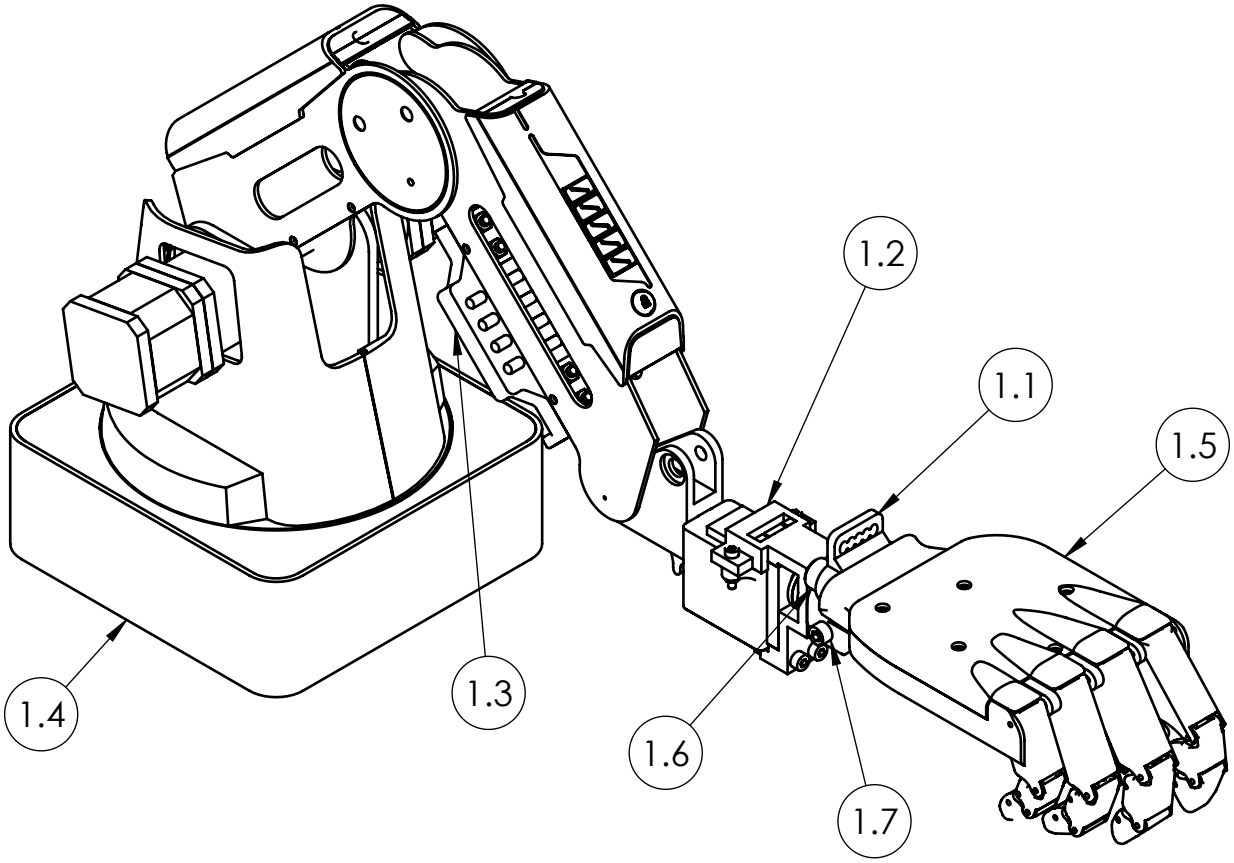
Descripción	Precio/hora (€)	Tiempo (h)	Coste (€)
Ingeniero técnico	11	450	4950
Técnico de laboratorio	9	5	45
		Total	4995

- **Presupuesto**

Una vez calculados los costes en el apartado anterior se puede obtener el presupuesto. Al valor obtenido se le suman los gastos generales debidos al mantenimiento de las instalaciones y equipos utilizados, así como los gastos en luz, agua o teléfono durante la elaboración del proyecto.

Concepto	Importe (€)
Amortización de los equipos	90,27
Material	4,6
Software	615,97
Mano de obra	4995
Subtotal	5705,84
Gastos generales (20%)	1141,17
Total	**6.847,01**

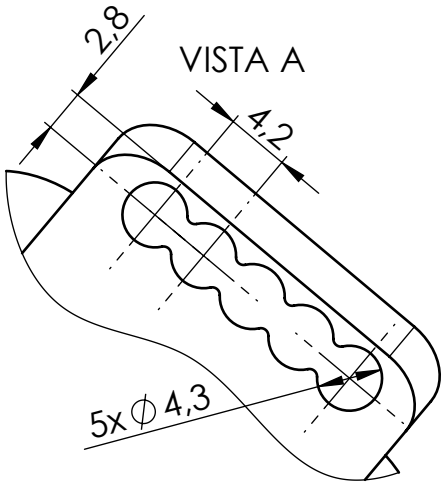
4. PLANOS

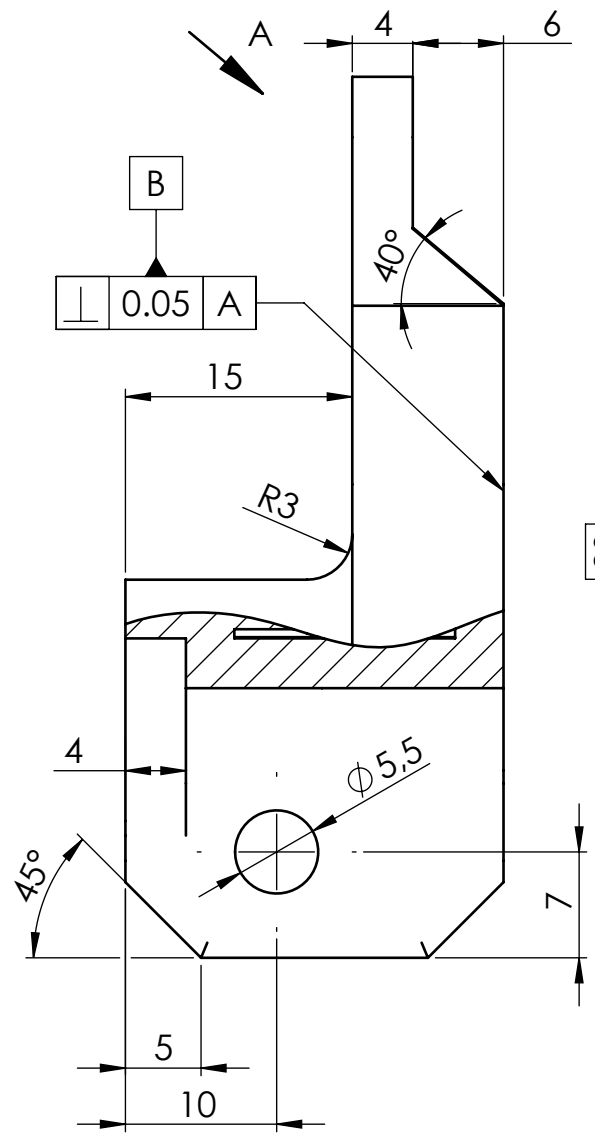


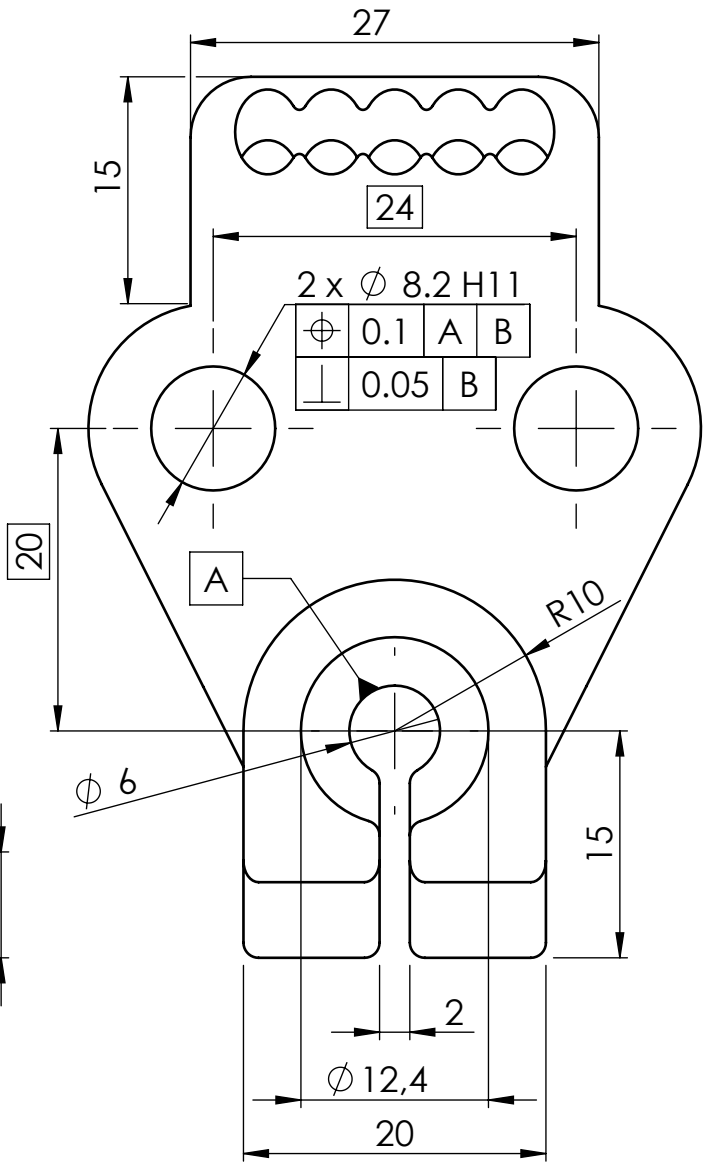
MARCA	DENOMINACIÓN	CANTIDAD	OBSERVACIONES
1.8	ISO - 4035 - M5 - N	1	Comercial
1.7	ISO 4762 M5 x 25 - 25N	1	Comercial
1.6	ISO 4762 M8 x 25 - 25N	2	Comercial
1.5	Mano protésica	1	Laboratorio
1.4	Dobot Magician	1	Comercial
1.3	Subensamblaje soporte actuadores	1	Plano 1.3
1.2	Subensamblaje servo	1	Plano 1.2
1.1	Acoplador mano	1	Plano 1.1

	Observaciones:	Unidad dim:	Escala:	Método de representación:
		mm		

	Creado por:	Revisado por:	Tipo de documento:			
UNIVERSITAT JAUME·I	Adrián Martínez	Adrián Martínez	Dibujo de diseño			
	Título: Ensamblaje robot		Nº de plano: Plano 1			
			Formato: A4	Idioma: es	Fecha: 10/12/2019	Hoja: 1/1



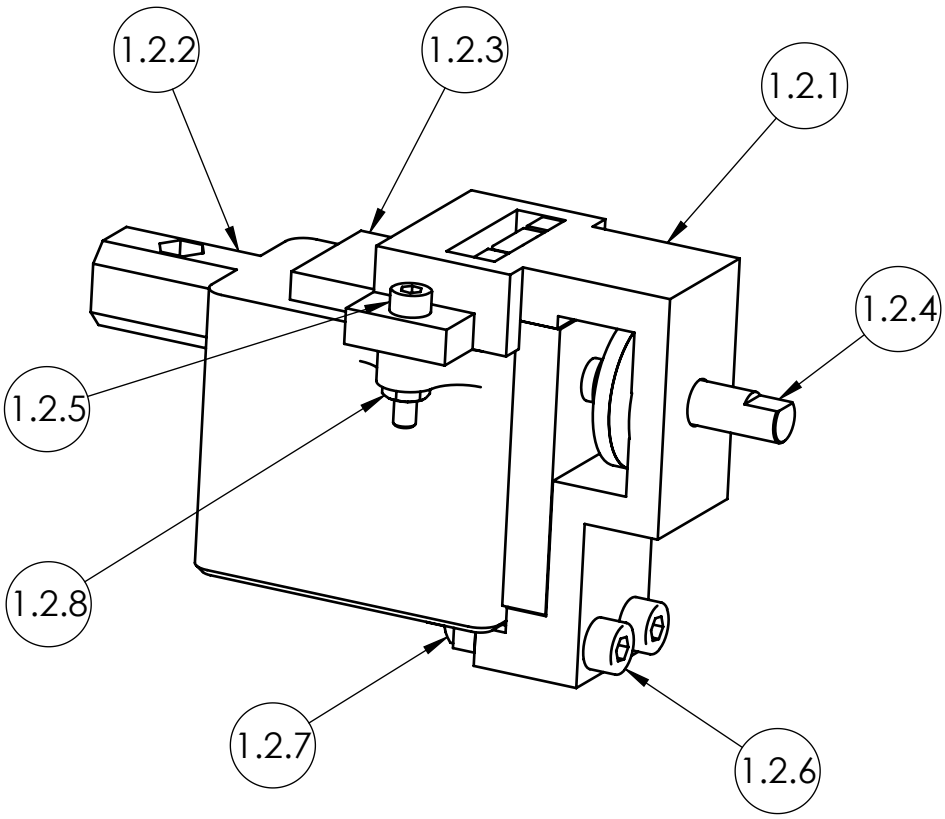




	Observaciones: Tolerancia general según ISO2768-mk Pieza hueca con espesor de pared de 3,2 mm	Unidad dim: mm	Escala: 2:1	Método de representación:

UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño
	Título: Acoplador mano	Nº de plano: Plano 1.1	

Formato:	Idioma:	Fecha:	Hoja:
A4	es	10/12/2019	1/1

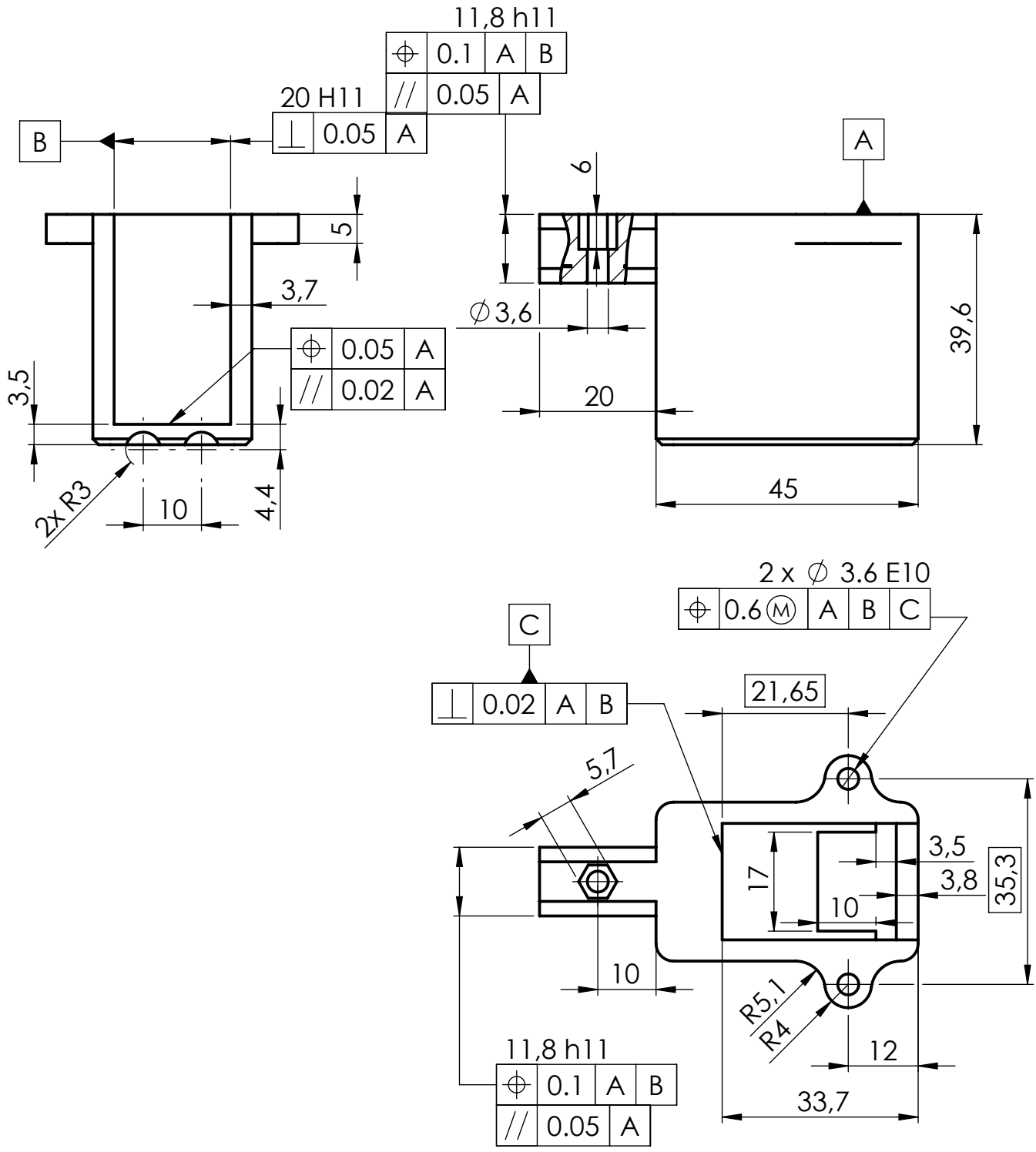


MARCA	DENOMINACIÓN	CANTIDAD	OBSERVACIONES
1.2.8	ISO - 4035 - M3 - N	2	Comercial
1.2.7	ISO - 4035 - M4 - N	2	Comercial
1.2.6	ISO 4762 M4 x 25 - 25N	2	Comercial
1.2.5	ISO 4762 M3 x 16 - 16N	2	Comercial
1.2.4	Eje servo	1	Dobot Magician
1.2.3	Servo	1	Dobot Magician
1.2.2	Soporte servo	1	Plano 1.2.2
1.2.1	Puente servo	1	Plano 1.2.1

	Observaciones:	Unidad dim: mm	Escala:	Método de representación:

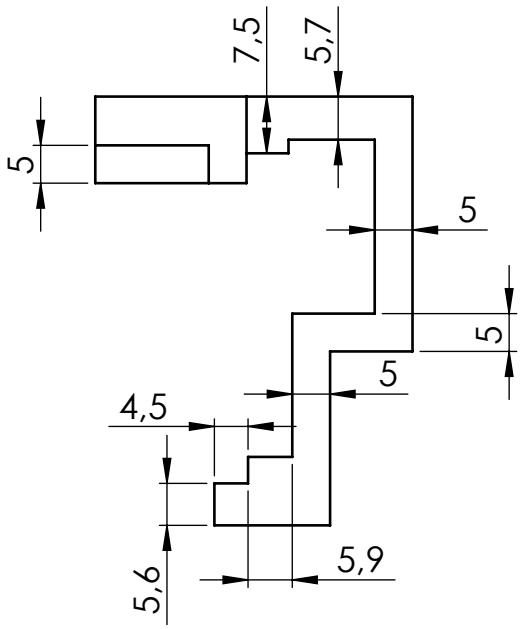
UNIVERSITAT JAUME·I

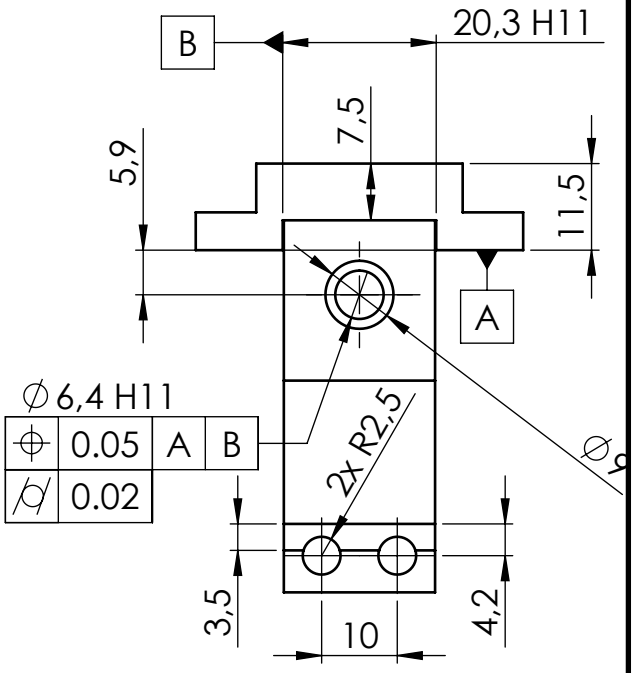
Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño		
Título: Subensamblaje servo		Nº de plano: Plano 1.2		
		Formato: A4	Idioma: es	Fecha: 10/12/2019	Hoja: 1/1

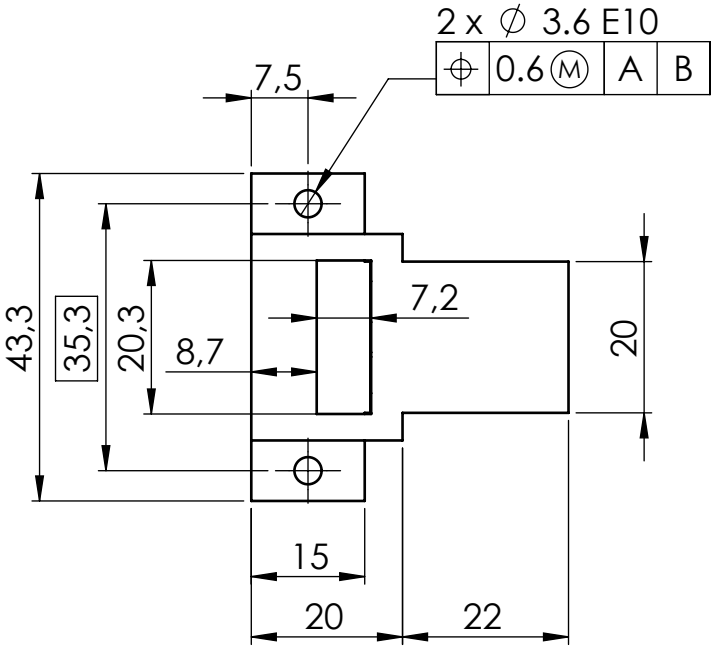


	Observaciones: Tolerancia general según ISO2768-mk Pieza hueca con espesor de pared de 2,8 mm	Unidad dim: mm	Escala: 1:1	Método de representación:

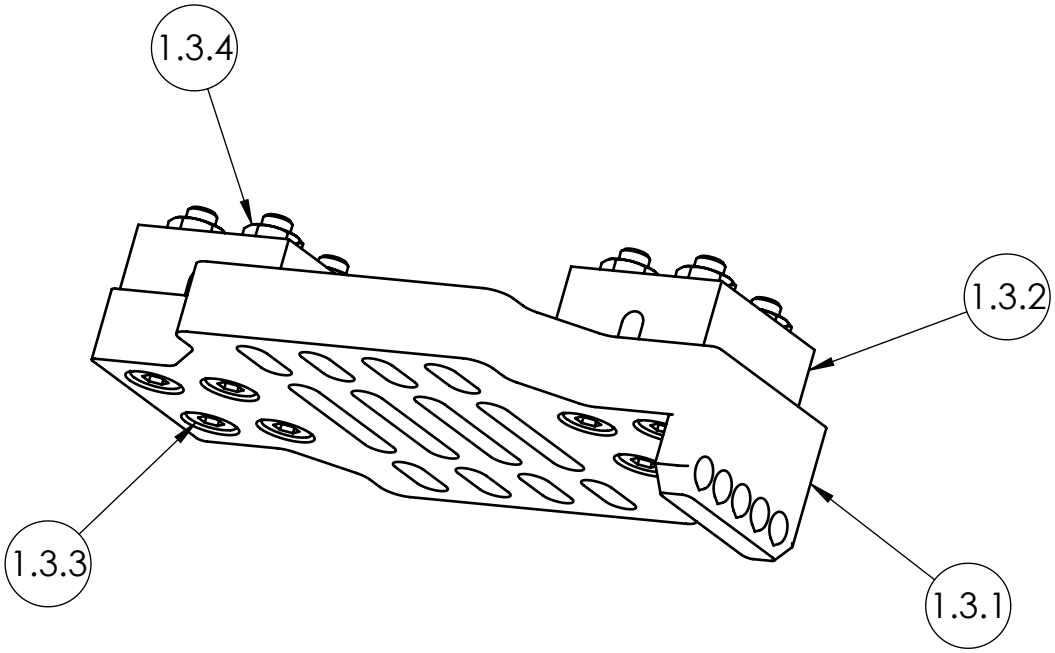
UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño		
	Título: Soporte servo		Nº de plano: Plano 1.2.1		
			Formato: A4	Idioma: es	Fecha: 10/12/2019	Hoja: 1/1







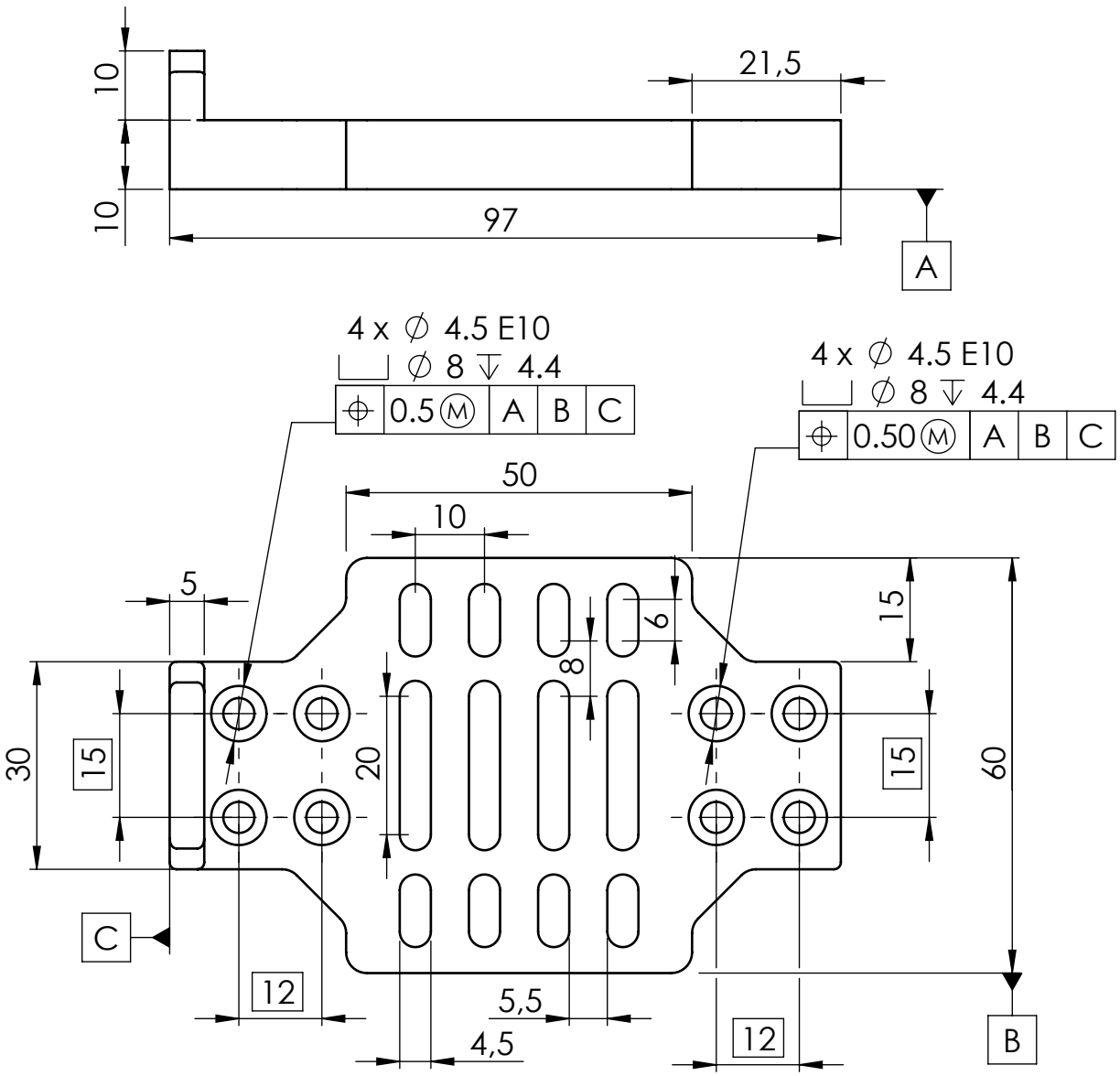
	Observaciones: Tolerancia general según ISO2768-mk	Unidad dim: mm	Escala: 1:1	Método de representación:
UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño	
	Título: Puente servo		Nº de plano: Plano 1.2.2	
			Formato: A4 / Idioma: es / Fecha: 10/12/2019 / Hoja: 1/1	



MARCA	DENOMINACIÓN	CANTIDAD	OBSERVACIONES
1.3.4	ISO - 4035 - M4 - N	8	Comercial
1.3.3	ISO 4762 M4 x 20 - 20N	8	Comercial
1.3.2	Fijador soporte actuadores	2	Plano 1.3.2
1.3.1	Base soporte actuadores	1	Plano 1.3.1

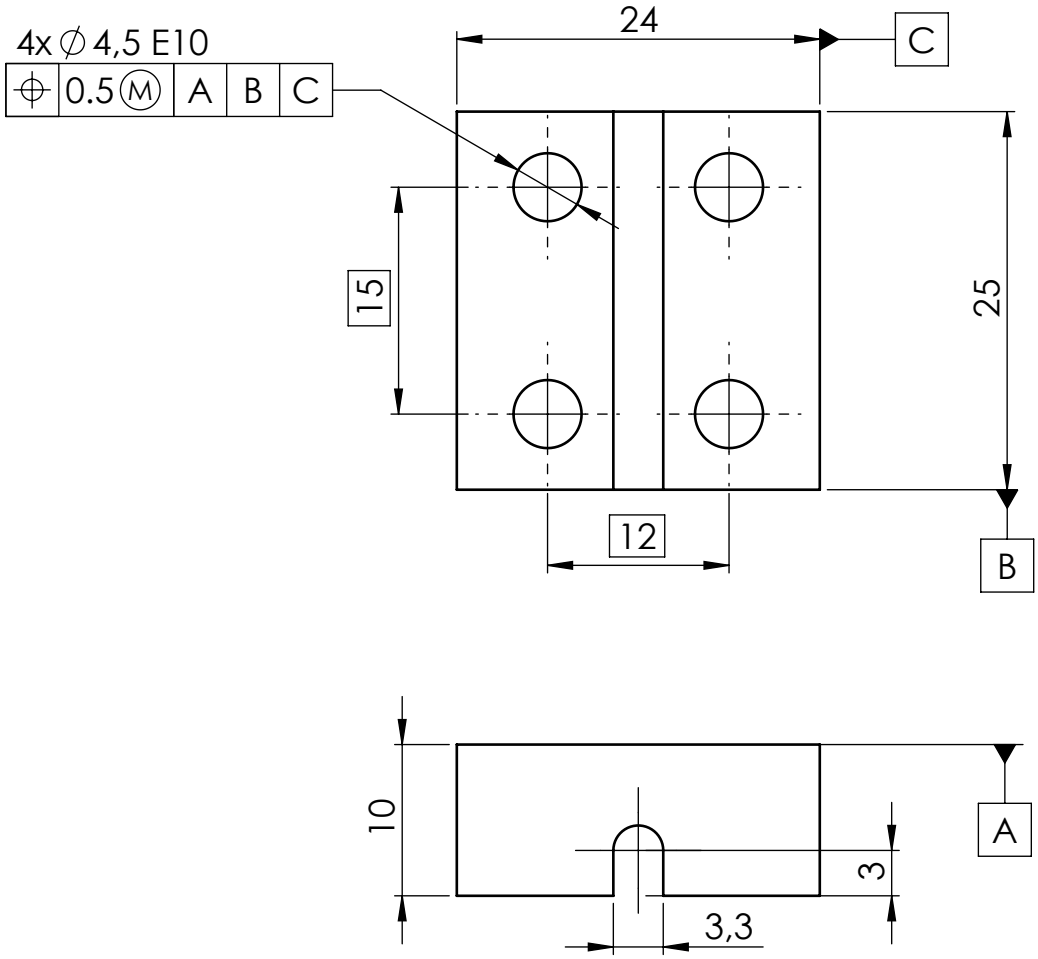
	Observaciones:	Unidad dim: mm	Escala:	Método de representación:

UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño		
	Título: Subensamblaje soporte actuadores		Nº de plano: Plano 1.3		
			Formato: A4	Idioma: es	Fecha: 10/12/2019	Hoja: 1/1



	Observaciones: Tolerancia general según ISO2768-mk Pieza hueca con espesor de pared de 3,2 mm	Unidad dim: mm	Escala: 1:1	Método de representación:

UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño		
	Título: Base soporte actuadores		Nº de plano: Plano 1.3.1		
			Formato: A4	Idioma: es	Fecha: 10/12/2019	Hoja: 1/1



	Observaciones: Tolerancia general según ISO2768-mk Pieza hueca con espesor de pared de 3,2 mm	Unidad dim: mm	Escala: 2:1	Método de representación:

UNIVERSITAT JAUME·I	Creado por: Adrián Martínez	Revisado por: Adrián Martínez	Tipo de documento: Dibujo de diseño
	Título: Fijador soporte actuadores	Nº de plano: Plano 1.3.2	

Formato:	Idioma:	Fecha:	Hoja:
A4	es	10/12/2019	1/1